

# ETSI TS 102 812 V1.1.1 (2001-11)

---

*Technical Specification*

## Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.1

---

European Broadcasting Union



Union Européenne de Radio-Télévision

EBU·UER  
**DVB**  
Digital Video  
Broadcasting



---

**Reference**

DTS/JTC-DVB-115

---

**Keywords**broadcasting, data, digital, DVB, MPEG,  
terrestrial, TV, video**ETSI**

---

650 Route des Lucioles  
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C  
Association à but non lucratif enregistré à  
Sous-Préfecture de Grasse (06) N° 7803/88

---

**Important notice**

Individual copies of the present document can be downloaded from:

<http://www.etsi.org>

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at

<http://portal.etsi.org/tb/status/status.asp>

If you find errors in the present document, send your comment to:  
editor@etsi.fr

---

**Copyright Notification**

No part may be reproduced except as authorized by written permission.  
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute, EBU 2001.  
All rights reserved.

# Contents

Intellectual Property Rights .....	34
Foreword .....	34
0 Introduction .....	34
0.1 Purpose .....	34
0.2 Application Areas .....	35
0.3 Profiles .....	35
1 Scope .....	36
2 References .....	36
3 Definitions and abbreviations .....	42
3.1 Definitions .....	42
3.2 Abbreviations .....	45
4 Conventions .....	47
5 Basic Architecture .....	48
5.1 Context .....	48
5.2 Architecture .....	49
5.2.1 Resources .....	49
5.2.2 System software .....	49
5.2.2.1 Application Manager .....	49
5.2.3 Application .....	49
5.3 Interfaces Between an MHP Application and the MHP System .....	51
5.4 Plug-ins .....	52
5.4.1 Security Model .....	53
6 Transport Protocols .....	54
6.1 Introduction .....	54
6.2 Broadcast Channel Protocols .....	54
6.2.1 MPEG-2 Transport Stream .....	55
6.2.2 MPEG-2 Sections .....	55
6.2.3 DSM-CC Private Data .....	55
6.2.4 DSM-CC Data Carousel .....	55
6.2.5 DSM-CC User-to-User Object Carousel .....	55
6.2.5.1 DVB-J class files .....	55
6.2.5.2 DVB-HTML document files .....	55
6.2.5.3 Loss of Carousel Behaviour .....	56
6.2.6 DVB Multiprotocol Encapsulation .....	56
6.2.7 Internet Protocol (IP) .....	56
6.2.8 User Datagram Protocol (UDP) .....	56
6.2.9 DVB Service Information .....	56
6.3 Interaction Channel Protocols .....	57
6.3.1 Network Dependent Protocols .....	57
6.3.2 Internet Protocol (IP) .....	57
6.3.3 Transmission Control Protocol (TCP) .....	57
6.3.4 UNO-RPC .....	57
6.3.5 UNO-CDR .....	57
6.3.6 DCM-CC User to User .....	57
6.3.7 Hypertext Transfer Protocol (HTTP) .....	58
6.3.7.1 HTTP 1.1 .....	58
6.3.7.2 MHP profile of HTTP 1.0 .....	58
6.3.7.2.1 HTTP 1.0 persistent connections .....	58
6.3.7.2.2 The Keep-Alive Header .....	58
6.3.7.2.3 MHP and proxies .....	58
6.3.7.2.4 Version compatibility .....	58
6.3.7.3 HTTPS .....	59

6.3.8	Service Specific . . . . .	59
6.3.9	User Datagram Protocol (UDP). . . . .	59
6.4	Transport protocols for application loading over the interaction channel . . . . .	60
6.4.1	File system implemented only via the interaction channel . . . . .	60
6.4.1.1	File system logical structure . . . . .	60
6.4.1.2	File transfer . . . . .	61
6.4.1.3	Class encoding . . . . .	61
6.4.1.4	Directory listing in this file system . . . . .	61
6.4.2	Hybrid between broadcast stream and interaction channel . . . . .	62
6.4.2.1	File transfer . . . . .	62
6.4.2.1.1	Broadcast file delivery . . . . .	62
6.4.2.1.2	Interaction channel delivery . . . . .	62
6.4.2.1.3	HTTPProfileBody . . . . .	62
6.4.2.2	Class encoding . . . . .	63
7	Content formats . . . . .	64
7.1	Static formats . . . . .	64
7.1.1	Bitmap image formats . . . . .	64
7.1.1.1	Image encoding restrictions . . . . .	64
7.1.1.2	JPEG . . . . .	64
7.1.1.3	PNG . . . . .	64
7.1.1.4	GIF . . . . .	64
7.1.2	MPEG-2 I-Frames . . . . .	64
7.1.3	MPEG-2 Video "drips" . . . . .	64
7.1.4	Monomedia format for audio clips . . . . .	66
7.1.5	Monomedia format for text . . . . .	66
7.1.5.1	Built-in character set . . . . .	66
7.2	Broadcast streaming formats . . . . .	66
7.2.1	Audio . . . . .	66
7.2.2	Video . . . . .	66
7.2.3	Subtitles . . . . .	66
7.2.3.1	DVB Subtitles . . . . .	66
7.2.3.2	Teletext . . . . .	67
7.3	Resident fonts . . . . .	67
7.4	Downloadable Fonts . . . . .	67
7.5	Colour Representation . . . . .	67
7.5.1	Background (informative) . . . . .	67
7.5.2	Specification . . . . .	68
7.5.2.1	The sRGB Reference Viewing Environment . . . . .	68
7.5.2.2	Colourimetric Definitions and Encodings . . . . .	68
7.6	MIME Types . . . . .	70
7.6.1	Rationale . . . . .	70
8	DVB-HTML . . . . .	71
8.1	Introduction . . . . .	71
8.1.1	Application Area . . . . .	71
8.1.2	Profiles . . . . .	72
8.2	Architecture . . . . .	72
8.2.1	Context . . . . .	72
8.2.2	Integration Aspects . . . . .	72
8.2.2.1	Accessing DVB-J from ECMAScript . . . . .	72
8.2.2.2	Implementation of user agents via plug-ins . . . . .	72
8.3	Application Format . . . . .	73
8.3.1	Basic Considerations . . . . .	73
8.3.2	Approach to Subsetting . . . . .	73
8.4	XML . . . . .	74
8.5	DVB Mark-up Language (DVB-HTML) . . . . .	74
8.5.1	Conformance considerations . . . . .	74
8.5.1.1	Document conformance . . . . .	74
8.5.1.1.1	General rules . . . . .	74



8.5.1.1.2	Invalid but conformant documents . . . . .	75
8.5.1.2	DVB-HTML user agent conformance . . . . .	75
8.5.1.2.1	Error handling . . . . .	77
8.5.1.2.2	Handling of invalid but conformant documents . . . . .	77
8.5.2	Set of modules required by this specification . . . . .	77
8.5.3	Semantics for modules . . . . .	78
8.5.3.1	XHTML modules . . . . .	78
8.5.3.1.1	Structure . . . . .	78
8.5.3.1.2	Text . . . . .	78
8.5.3.1.3	Hypertext . . . . .	78
8.5.3.1.4	Presentation . . . . .	78
8.5.3.1.5	Forms . . . . .	79
8.5.3.1.6	Client-side Image Map . . . . .	79
8.5.3.1.7	Image . . . . .	79
8.5.3.1.8	Object . . . . .	79
8.5.3.1.9	Frames . . . . .	79
8.5.3.1.10	Target . . . . .	79
8.5.3.1.11	Iframes . . . . .	79
8.5.3.1.12	Metainformation . . . . .	80
8.5.3.1.13	Scripting . . . . .	80
8.5.3.1.14	Link . . . . .	80
8.5.3.1.15	Base . . . . .	80
8.5.3.2	XHTML attributes . . . . .	80
8.5.3.2.1	Longdesc, alt and cite attributes . . . . .	80
8.5.3.2.2	Accesskey attribute . . . . .	80
8.5.3.3	DVB-HTML modules . . . . .	81
8.5.3.3.1	DVB Intrinsic events . . . . .	81
8.6	Media Types . . . . .	83
8.6.1	Uses of MIME media types . . . . .	83
8.6.2	MIME media type use restrictions . . . . .	84
8.6.3	Semantics of media type . . . . .	86
8.6.4	Frame content . . . . .	86
8.6.5	Application content . . . . .	86
8.6.5.1	When referenced via an AIT locator . . . . .	86
8.6.5.2	When not referenced via an AIT locator . . . . .	87
8.6.6	Relative linking . . . . .	87
8.6.7	MPEG Audio . . . . .	87
8.6.7.1	Resources of indefinite duration . . . . .	87
8.6.7.1.1	Relation to document events . . . . .	87
8.6.7.2	Resources of definite duration . . . . .	88
8.6.7.2.1	Relation to document events . . . . .	88
8.6.8	MPEG Video . . . . .	88
8.6.8.1	Video Resources of indefinite duration . . . . .	88
8.6.8.1.1	Relation to document events . . . . .	89
8.6.8.2	Resources of definite duration . . . . .	89
8.6.9	DVB Services . . . . .	89
8.6.10	Graphics content . . . . .	89
8.6.11	Script content . . . . .	89
8.6.12	Style sheet content . . . . .	89
8.6.13	HTTP(S) URLs . . . . .	90
8.6.14	CSS Properties . . . . .	90
8.6.14.1	Sources of MIME media type use points . . . . .	90
8.6.14.2	MIME media type use restrictions . . . . .	90
8.6.15	Generated Content . . . . .	91
8.6.16	Graphics styling . . . . .	91
8.6.17	Video Styling . . . . .	91
8.6.18	DVB Service styling . . . . .	91
8.7	Synchronization . . . . .	92
8.7.1	Triggers Overview . . . . .	92

8.7.1.1	Transport of triggers .....	92
8.7.1.2	Application registration and reception. ....	92
8.7.1.3	Binding to DSM-CC Stream events. ....	92
8.7.2	Trigger Events .....	93
8.7.2.1	Converting stream events into DOM events .....	93
8.7.2.2	Event Factory File definition .....	94
8.7.2.2.1	Syntax .....	94
8.7.2.2.2	Element semantics .....	95
8.7.2.2.3	Attributes semantics .....	95
8.7.2.3	Default Event Factory Element .....	97
8.7.2.4	Default Event Factory File .....	97
8.7.2.5	Worked example .....	97
8.7.2.6	System events .....	98
8.7.2.6.1	dvb.start event .....	98
8.7.2.6.2	dvb.page event .....	99
8.7.3	Binding the event factory file to the application .....	99
8.7.3.1	Syntax of event linkage file .....	100
8.7.3.2	Semantics of event linkage file .....	100
8.7.3.3	Example .....	100
8.7.3.4	Name and location of linkage file .....	101
8.7.4	Default Trigger Mechanism .....	101
8.8	CSS .....	103
8.8.1	Summary of CSS profiling for MHP .....	103
8.8.2	MHP profile of CSS data types .....	103
8.8.3	MHP profile of CSS @ rules .....	103
8.8.4	MHP profile of CSS media types .....	104
8.8.4.1	"screen" media type .....	104
8.8.4.2	'dvb-tv' media type .....	104
8.8.4.2.1	Additional Properties of "dvb-tv" media type .....	104
8.8.4.2.2	Policy rules .....	105
8.8.4.3	Clarifications on support of paged properties .....	105
8.8.5	Graphics and video integration .....	105
8.8.5.1	General recap of the MHP graphics .....	105
8.8.5.1.1	Input video space .....	105
8.8.5.1.2	Device space .....	105
8.8.5.1.3	Normalised space .....	105
8.8.5.1.4	Colour .....	105
8.8.5.2	Coordinate spaces .....	106
8.8.5.2.1	Screen coordinates .....	106
8.8.5.2.2	Pixel coordinates .....	106
8.8.5.2.3	Video coordinates .....	106
8.8.5.3	How to define the initial containing block .....	107
8.8.5.3.1	Problem .....	107
8.8.5.3.2	The @viewport rule .....	107
8.8.5.3.3	Establishing a viewport .....	108
8.8.5.3.4	Pseudo classes .....	112
8.8.5.4	Cascading .....	113
8.8.5.5	How to discover where the video is .....	113
8.8.5.5.1	The area property .....	113
8.8.5.6	Placing content in relation to video .....	114
8.8.5.6.1	Definition of boxes .....	115
8.8.5.6.2	Definition of pel areas in the video .....	115
8.8.5.7	Placing video within the presentation .....	115
8.8.5.8	Box Layout .....	115
8.8.5.8.1	Video Boxes .....	115
8.8.5.9	DOM Access to CSS .....	116
8.8.5.10	Focus traversal and short-cuts .....	116
8.8.6	Font selection .....	117
8.8.6.1	Restrictions on "src" descriptor .....	118

8.8.7	Font specification .....	118
8.8.8	Default behaviour .....	118
8.8.8.1	Default style sheet font rules .....	119
8.8.8.1.1	Extending the simple rule. ....	119
8.8.8.1.2	Fallback for italic, small caps and font stretch. ....	119
8.9	Xlet integration .....	120
8.9.1	Object element. ....	120
8.9.2	Param element. ....	121
8.9.3	Example. ....	121
8.10	Scripting .....	122
8.10.1	DOM 2 binding .....	122
8.10.2	Interface between ECMAScript and DVB-J .....	122
8.10.2.1	ECMAScript APIs for accessing DVB-J. ....	122
8.10.2.2	Inter-Xlet and Xlet-ECMAScript Communication via org.dvb.ixc .....	122
8.10.2.3	Security .....	123
8.10.2.4	Implicit Method Selection .....	123
8.10.2.5	Explicit Method Selection .....	123
8.10.2.6	Static Method Invocation. ....	123
8.10.2.7	Method Signature Matching .....	123
8.10.2.8	New ECMAScript Object Types .....	124
8.10.2.9	Type Conversion (ECMAScript to DVB-J). ....	124
8.10.2.10	Subclassing and Interface Instance Creation .....	126
8.10.2.11	Type Conversion (DVB-J to ECMAScript). ....	126
8.10.2.12	Catching DVB-J Exceptions in ECMAScript .....	127
8.11	Document Object Model (DOM) .....	128
8.11.1	DOM Level 2 Events. ....	128
8.11.1.1	Fundamental interfaces .....	128
8.11.1.2	Event interfaces .....	128
8.11.2	DVB Events DOM module .....	129
8.11.2.1	Key events .....	129
8.11.2.2	Lifecycle events .....	129
8.11.2.2.1	Interface DVBLifecycleEvent .....	129
8.11.2.2.2	Event definitions .....	130
8.11.2.2.3	State transition summary .....	132
8.11.2.3	Additional DVB Events .....	132
8.11.2.3.1	Trigger events. ....	132
8.11.2.3.2	DVBDOMStable event .....	132
8.11.2.3.3	DVB-HTML events .....	133
8.11.3	DVB Key events DOM module .....	133
8.11.3.1	Interface DVBKeyEvent .....	133
8.11.3.1.1	IDL Definition .....	134
8.11.3.1.2	Attributes .....	135
8.11.3.1.3	Methods .....	135
8.11.4	DVB-HTML DOM module. ....	135
8.11.4.1	Conformance .....	135
8.11.4.2	Differences from W3C DOM Level 1 HTML interfaces .....	136
8.11.4.3	Extensions .....	136
8.11.4.3.1	Enumerations .....	136
8.11.4.3.2	Initial and current values of form controls. ....	136
8.11.4.4	System aspects .....	137
8.11.4.4.1	Access to the document .....	137
8.11.4.4.2	DOM DVB-HTML module .....	137
8.11.4.4.3	DOM modification .....	137
8.11.4.5	Miscellaneous interfaces .....	137
8.11.4.5.1	DVB-HTMLCollection Interface. ....	137
8.11.4.5.2	DVBHTMLDocument Interface .....	138
8.11.4.6	DVB-HTML element related interfaces .....	140
8.11.4.6.1	DVBHTMLElement Interface .....	140
8.11.4.6.2	DVBHTMLAnchorElement Interface .....	141

8.11.4.6.3	DVBHTMLMapElement Interface	141
8.11.4.6.4	DVBHTMLAreaElement Interface	142
8.11.4.6.5	DVBHTMLButtonElement Interface	142
8.11.4.6.6	DVBHTMLFormElement Interface	143
8.11.4.6.7	DVBHTMLFrameElement Interface	144
8.11.4.6.8	DVBHTMLFrameSetElement Interface	145
8.11.4.6.9	DVBHTMLIFrameElement Interface	145
8.11.4.6.10	DVBHTMLImageElement Interface	145
8.11.4.6.11	DVBHTMLObjectElement Interface	146
8.11.4.6.12	DVBHTMLInputElement Interface	147
8.11.4.6.13	DVBHTMLOptionElement Interface	149
8.11.4.6.14	DVBHTMLSelectElement Interface	150
8.11.4.6.15	DVBHTMLTextAreaElement Interface	151
8.11.5	DVB Exceptions	152
8.11.5.1	DVBException	153
8.11.5.1.1	IDL Definition	153
8.11.5.1.2	Defined Constants	153
8.11.6	Language bindings	153
8.11.6.1	ECMAScript Binding	153
8.11.6.2	Java Binding	153
8.11.7	DVB Environment object module	153
8.11.7.1	Free variables	153
8.11.7.2	Environmental host objects	154
8.11.7.2.1	Navigator Object	154
8.11.7.2.2	Window object	154
8.11.7.2.3	Location object	157
8.11.8	CSS Support	157
8.11.8.1	DVB CSS DOM module	157
8.11.8.1.1	DVBCSSInlineStyle	157
8.11.8.1.2	DVBCSSStyle	158
8.11.8.1.3	DVBCSSViewportRule	158
8.11.8.1.4	DVBCSSViewportProperties	158
8.12	Cookie support	161
8.12.1	DOM Cookie Interface	161
8.12.2	Cookie Storage and Lifetime	161
8.12.2.1	Cookie Storage Limits	161
8.12.2.2	Cookie Persistence	161
8.12.2.3	Privacy Considerations	162
8.12.3	Cookie Scoping	162
8.12.3.1	General Rules	162
8.12.3.2	Documents delivered via DSM-CC Object Carousel	162
8.12.3.3	Documents delivered via HTTP transport	162
8.12.4	HTTP Cookie Support	162
8.12.4.1	Background	162
8.12.4.2	Sending Cookies	162
8.12.4.3	Receiving Cookies	162
8.13	HTTP User Agent String Support	163
8.13.1	User agent strings	163
8.13.1.1	Current user agent-related strings	163
8.13.1.2	User agent string BNF	163
8.14	Security of DVB-HTML applications	164
8.14.1	Authentication of DVB-HTML files	164
8.14.2	Runtime code extension	164
8.14.2.1	Security principles	164
8.14.2.1.1	Uses of runtime code extension in ECMAScript	164
8.14.2.2	Extensions to ECMAScript for trusted executable code	165
8.14.2.2.1	Propagation of Internal (safe) vs. External (unsafe) strings	165
8.14.2.2.2	Modifying ECMA-262 to support Internal and External strings	165
8.14.2.3	Sources of Unsafe (external) strings	171

8.14.2.3.1	Sources within ECMAScript .....	171
8.14.2.3.2	Sources from Host Objects .....	171
8.14.2.4	Use of strings in RCEs .....	171
8.14.2.5	Mutation of Host Objects .....	172
8.14.3	Inter application security .....	172
8.14.3.1	Restrictions on DOM elements introduced for security .....	172
8.15	DVB-HTML permissions .....	172
8.15.1	Permissions for unsigned applications .....	173
8.15.1.1	java.awt.AWTPermission .....	173
8.15.1.2	java.net.SocketPermission: .....	173
8.15.1.3	java.util.PropertyPermission .....	173
8.15.1.4	java.lang.RuntimePermission .....	173
8.15.1.5	java.io.SerializablePermission .....	173
8.15.1.6	java.io.FilePermission .....	173
8.15.1.7	javax.tv.media.MediaSelectPermission .....	174
8.15.1.8	javax.tv.service.ReadPermission .....	174
8.15.1.9	javax.tv.service.selection.ServiceContextPermission .....	174
8.15.1.10	java.util.Locale.setDefault .....	175
8.15.1.11	org.dvb.security.PrivilegedRCEPermission .....	175
8.15.2	Additional Permissions for signed applications .....	175
8.15.2.1	java.util.PropertyPermission .....	175
8.15.2.2	java.io.FilePermission .....	175
8.15.2.3	org.dvb.net.ca.CAPermission .....	176
8.15.2.4	org.dvb.application.AppsControlPermission .....	176
8.15.2.5	org.dvb.net.rc.RCPermission .....	177
8.15.2.6	org.dvb.net.tuning.TunerPermission .....	177
8.15.2.7	javax.tv.service.selection.SelectPermission .....	178
8.15.2.8	org.dvb.user.UserPreferencePermission .....	178
8.15.2.9	java.net.SocketPermission .....	178
8.15.2.10	org.dvb.media.DripFeedPermission .....	178
8.15.2.11	org.dvb.security.PrivilegedRCEPermission .....	179
8.15.2.12	org.dvb.application.storage.ApplicationStoragePermission .....	179
8.15.2.13	org.dvb.smartcard.SmartCardPermission .....	179
8.16	Miscellaneous .....	179
8.16.1	Date Values .....	179
8.16.1.1	Syntax .....	179
8.16.2	Clock values .....	179
8.16.2.1	Syntax .....	179
8.16.2.2	Offset values .....	180
8.16.3	Unrealisable locators .....	180
8.16.3.0.1	Presentation of Locators in DVB HTML .....	180
8.16.4	Relation to HTTP and HTTPS .....	181
8.16.5	DVB-HTML specific locators .....	181
8.16.5.1	Extended DVB locator .....	181
8.16.5.1.1	Extended DVB locator syntax .....	181
8.16.5.1.2	TV locators .....	181
8.16.5.1.3	Application locator .....	182
8.16.5.1.4	AIT locators .....	182
8.16.5.2	Exit locator .....	182
8.16.6	Domain .....	182
9	Application model .....	183
9.1	Broadcast MHP applications .....	183
9.1.1	Basic lifecycle control .....	183
9.1.2	Starting applications .....	184
9.1.3	Support for execution of multiple simultaneous applications .....	184
9.1.4	Stopping applications .....	184
9.1.4.1	A new service being selected replacing a previously selected one .....	184
9.1.4.2	The stopping of an application by another application .....	184
9.1.4.3	Changes in the application signalling to request a particular application be stopped .....	184

9.1.4.4	Stopping by the MHP terminal due to a shortage of resources . . . . .	185
9.1.5	Persistence of Applications Across Service Boundaries . . . . .	185
9.1.6	Management of autostarting . . . . .	185
9.1.7	When tuning is not service selection! . . . . .	186
9.1.8	MHP Applications and Service Selection . . . . .	186
9.1.9	Broadcast service related stored applications . . . . .	186
9.1.9.1	Version management . . . . .	187
9.2	DVB-J Model . . . . .	187
9.2.1	Starting DVB-J Applications . . . . .	187
9.2.2	Stopping a DVB-J Application . . . . .	187
9.2.3	DVB-J Application Lifecycle . . . . .	188
9.2.3.1	Introduction . . . . .	188
9.2.3.2	DVB-J Application Lifecycle State Machine . . . . .	188
9.2.4	Xlet API . . . . .	191
9.2.4.1	Xlet State Change Semantics . . . . .	192
9.2.4.2	Xlet state change requests . . . . .	192
9.2.5	Multiple application environment support . . . . .	192
9.2.5.1	Control of DVB-J applications by other DVB-J applications . . . . .	192
9.2.5.2	Input Focus management . . . . .	192
9.2.5.3	Other resources management . . . . .	193
9.2.5.4	VM implementation . . . . .	193
9.3	DVB-HTML Model . . . . .	193
9.3.1	The DVB-HTML Application . . . . .	193
9.3.1.1	DVB-HTML Application . . . . .	193
9.3.1.2	User agent . . . . .	193
9.3.1.3	DVB-HTML Actor . . . . .	193
9.3.1.4	Application boundary . . . . .	194
9.3.1.4.1	Regular Expression Syntax . . . . .	194
9.3.2	DVB-HTML Application Lifecycle . . . . .	195
9.3.2.1	Introduction . . . . .	195
9.3.2.2	Signalling . . . . .	195
9.3.2.3	Lifecycle control . . . . .	196
9.3.2.3.1	State diagram . . . . .	196
9.3.3	The State Model . . . . .	196
9.3.3.1	Loading . . . . .	197
9.3.3.1.1	Name . . . . .	197
9.3.3.1.2	Entry actions . . . . .	197
9.3.3.1.3	Activities . . . . .	197
9.3.3.1.4	Resources . . . . .	197
9.3.3.1.5	Transitions . . . . .	197
9.3.3.1.6	Comment . . . . .	197
9.3.3.2	Active . . . . .	197
9.3.3.2.1	Name . . . . .	197
9.3.3.2.2	Activities . . . . .	197
9.3.3.2.3	Entry actions . . . . .	197
9.3.3.2.4	Resources . . . . .	197
9.3.3.2.5	Transitions . . . . .	198
9.3.3.2.6	Comment . . . . .	198
9.3.3.3	Paused . . . . .	198
9.3.3.3.1	Name . . . . .	198
9.3.3.3.2	Activities . . . . .	198
9.3.3.3.3	Resources . . . . .	198
9.3.3.3.4	Transitions . . . . .	198
9.3.3.3.5	Comment . . . . .	198
9.3.3.4	Destroyed . . . . .	199
9.3.3.4.1	Name . . . . .	199
9.3.3.4.2	Activities . . . . .	199
9.3.3.4.3	Resources . . . . .	199
9.3.3.4.4	Transitions: . . . . .	199

9.3.3.4.5	Comment .....	199
9.3.3.5	Killed .....	199
9.3.3.5.1	Name .....	199
9.3.3.5.2	Entry actions .....	199
9.3.3.5.3	Activities .....	199
9.3.3.5.4	Resources .....	199
9.3.3.5.5	Transitions .....	199
9.3.3.5.6	Comment .....	199
9.4	Application activity events .....	200
9.4.1	Event queue handling .....	202
9.5	Inter application resource management .....	202
9.6	Life cycle of Xlets embedded in DVB-HTML .....	203
9.6.1	Starting embedded Xlets .....	203
9.6.2	Termination .....	203
9.6.3	General issues .....	203
9.7	Services and applications not related to conventional DVB services .....	204
9.7.1	Applications loaded from the interaction channel .....	204
9.7.2	Stored services .....	204
9.7.3	DVB-J Model .....	206
9.7.4	Common behaviour .....	206
9.8	Lifecycle of internet access applications .....	206
9.8.1	General issues .....	206
9.8.2	Starting internet access applications from MHP applications .....	206
9.8.3	Starting MHP applications from internet access applications .....	207
9.9	Plug-ins .....	207
10	Application Signalling .....	208
10.1	Introduction .....	208
10.1.1	Summary of common signalling .....	208
10.1.2	Summary of additional signalling for DVB-J applications .....	208
10.1.3	Summary of additional signalling for DVB-HTML applications .....	208
10.1.4	Summary of additional signalling for applications carried via OC .....	208
10.1.5	Summary of additional signalling for applications carried via IP .....	209
10.1.6	How to add a new scheme (informative) .....	209
10.1.7	Service information .....	209
10.2	Program Specific Information .....	209
10.2.1	Application signalling stream .....	209
10.2.2	Data broadcast streams .....	209
10.3	Notation .....	210
10.3.1	reserved .....	210
10.3.2	reserved_future_use .....	210
10.4	Application Information Table .....	210
10.4.1	Data errors .....	210
10.4.2	AIT transmission and monitoring .....	210
10.4.3	Optimised AIT signalling .....	211
10.4.4	Visibility of AIT .....	211
10.4.5	Definition of sub-table for the AIT .....	211
10.4.6	Syntax of the AIT .....	211
10.4.7	Use of private descriptors in the AIT .....	213
10.4.8	Text encoding in AIT .....	213
10.4.9	AIT file .....	213
10.4.9.1	Syntax .....	213
10.4.9.2	Syntactic restrictions .....	213
10.4.9.2.1	Transport protocols .....	213
10.4.9.3	Semantics .....	213
10.4.9.4	MIME type .....	214
10.5	Application identification .....	214
10.5.1	Encoding .....	214
10.5.2	Effects on life cycle .....	215

10.5.3	Authentication of application identification	215
10.6	Control of application life cycle	215
10.6.1	Entering and leaving the domain of an application	215
10.6.2	Dynamic control of the application life cycle	215
10.6.2.1	DVB-J	216
10.6.2.2	DVB-HTML	216
10.7	Generic descriptors	217
10.7.1	Application Signalling Descriptor	217
10.7.2	Data broadcast id descriptor	217
10.7.2.1	Generic descriptor	218
10.7.2.2	MHP data broadcast id descriptor	218
10.7.3	Application descriptor	219
10.7.4	User information descriptors	220
10.7.4.1	Application name descriptor	221
10.7.4.2	Application icons descriptor	221
10.7.5	External application authorisation descriptor	223
10.8	Transport protocol descriptors	223
10.8.1	Transport protocol descriptor	223
10.8.1.1	Transport via OC	224
10.8.1.2	Transport via IP	225
10.8.1.3	Transport via interaction channel	226
10.8.2	IP Routing Descriptors	226
10.8.2.1	Routing Descriptor IPv4	227
10.8.2.2	Routing Descriptor IPv6	227
10.8.3	Pre-fetch signalling	228
10.8.3.1	Introduction	228
10.8.3.2	Pre-fetch descriptor	228
10.8.3.3	DII location descriptor	229
10.9	DVB-J specific descriptors	230
10.9.1	DVB-J application descriptor	230
10.9.2	DVB-J application location descriptor	230
10.10	DVB-HTML Specific descriptors	231
10.10.1	DVB-HTML application descriptor	231
10.10.2	DVB-HTML application location descriptor	232
10.10.2.1	Example	232
10.10.2.2	Application Entry Point	232
10.10.3	DVB-HTML application boundary descriptor	233
10.11	Constant values	234
10.12	Service Information	235
10.12.1	Service identifier descriptor	235
10.13	Plug-in signalling	236
10.13.1	Native signalling scenario	236
10.13.2	MHP signalling scenario	236
10.13.3	delegated application descriptor	236
10.13.4	Plug-in descriptor	237
10.14	Stored applications	238
10.14.1	Use of signalling defined in MHP 1.0	238
10.14.1.1	Stored broadcast service related applications	238
10.14.1.2	Stored stand-alone applications	238
10.14.2	Application storage descriptor	238
10.14.3	Application description file	239
10.14.3.1	Description	239
10.14.3.2	Application description file name and location	240
10.14.3.3	Syntax	240
10.14.3.4	Semantics	240
11	DVB-J Platform	241
11.1	The Virtual Machine	241
11.2	General issues	241
11.2.1	Basic Considerations	241



11.2.2	Approach to Subsetting . . . . .	241
11.2.3	Class Loading . . . . .	241
11.2.4	Unloading . . . . .	242
11.2.5	Event listeners . . . . .	242
11.2.6	Event model in DAVIC APIs . . . . .	242
11.2.7	Event model in DAVIC & DVB APIs . . . . .	242
11.2.8	Tuning as a side-effect. . . . .	242
11.2.9	Intra application media resource management . . . . .	242
11.2.10	Application thread priority . . . . .	242
11.3	Fundamental DVB-J APIs. . . . .	243
11.3.1	Java platform APIs . . . . .	243
11.3.1.1	java.lang package. . . . .	243
11.3.1.2	java.lang.reflect package . . . . .	244
11.3.1.3	java.util . . . . .	244
11.3.1.4	java.util.zip. . . . .	244
11.3.1.5	java.io. . . . .	244
11.3.1.6	java.net. . . . .	244
11.3.1.7	java.beans. . . . .	245
11.3.1.8	java.math . . . . .	245
11.3.1.9	java.text . . . . .	245
11.3.2	MHP platform APIs. . . . .	246
11.3.2.1	org.dvb.lang . . . . .	246
11.3.2.2	org.dvb.event . . . . .	246
11.4	Presentation APIs . . . . .	246
11.4.1	Graphical User Interface API . . . . .	246
11.4.1.1	The Core GUI API. . . . .	246
11.4.1.2	TV user interface . . . . .	248
11.4.1.3	Extended graphics . . . . .	249
11.4.1.4	Handling of input events . . . . .	249
11.4.2	Streamed Media API . . . . .	250
11.4.2.1	Framework of solution. . . . .	250
11.4.2.2	Clarifications . . . . .	250
11.4.2.3	Default media player behaviour. . . . .	251
11.4.2.4	Required controls for video drips . . . . .	251
11.4.2.5	Extensions to the Framework . . . . .	251
11.4.2.5.1	DVB specified extensions . . . . .	251
11.4.2.5.2	Extensions in org.davic . . . . .	251
11.4.2.5.3	Extensions in javax.tv . . . . .	252
11.4.2.5.4	Required controls for broadcast profiles . . . . .	252
11.4.2.5.5	Clarifications . . . . .	252
11.4.2.6	Restrictions on the Framework for Broadcast . . . . .	252
11.4.2.7	Intersection Between MediaSelectControl and SubtitlingLanguageControl / AudioLanguageControl. . . . .	253
11.4.2.8	Intersection between Streamed Media API and TV User Interface API . . . . .	254
11.4.2.8.1	Basic Principles . . . . .	254
11.4.2.8.2	TV Behaviour Control . . . . .	254
11.4.2.8.3	Application Behaviour Control . . . . .	254
11.4.2.8.4	Dynamic Behaviour . . . . .	255
11.4.2.8.5	Resource Management Details. . . . .	255
11.5	Data Access APIs . . . . .	255
11.5.1	Broadcast Transport Protocol Access API . . . . .	255
11.5.1.1	Constraints on the java.io.File methods for broadcast carousels. . . . .	256
11.5.1.2	Methods dealing with write access . . . . .	256
11.5.1.3	Behaviour following loss of a broadcast file system. . . . .	257
11.5.2	Support for Multicast IP over the Broadcast Channel. . . . .	257
11.5.3	Support for IP over the Return Channel . . . . .	258
11.5.4	MPEG-2 Section Filter API. . . . .	258
11.5.5	Mid-Level Communications API . . . . .	258

11.5.6	Persistent Storage API	258
11.6	Service Information and Selection APIs	259
11.6.1	DVB Service Information API	259
11.6.2	Service Selection API	259
11.6.3	Tuning API	259
11.6.4	Conditional Access API	260
11.6.5	Protocol Independent SI API	260
11.7	Common Infrastructure APIs	261
11.7.1	APIs to support DVB-J application lifecycle	261
11.7.1.1	Xlet properties	261
11.7.1.2	Actions for DVB-J applications to perform in their destroy method	261
11.7.2	Application discovery and launching APIs	261
11.7.3	Inter-Application and Inter-Xlet communication API	263
11.7.3.1	Remote Call Semantics	263
11.7.3.1.1	Objects Passed by Remote Reference	263
11.7.3.1.2	Objects Passed by Remote Copy	264
11.7.3.1.3	Classloading Considerations	265
11.7.3.1.4	Thread Usage	265
11.7.3.1.5	Garbage Collection of Remote Objects	265
11.7.4	Basic MPEG Concepts	265
11.7.5	Resource Notification	266
11.7.6	Content Referencing	266
11.7.7	Common Error Reporting	267
11.7.8	Plug-in APIs	267
11.8	Security	268
11.8.1	Basic Security	268
11.8.1.1	java.security	268
11.8.1.2	java.security.cert	269
11.8.1.3	Other classes	269
11.8.2	APIs to Support TLS / SSL Over the Return Channel	269
11.8.3	Additional permissions classes	269
11.8.4	General security issues	269
11.9	Other APIs	269
11.9.1	Timer Support	269
11.9.2	User Settings and Preferences API	270
11.9.3	Profile and version properties	270
11.9.3.1	Information on options	271
11.9.4	Non-CA smart card API	271
11.9.4.1	Framework	271
11.9.4.2	Clarifications	271
11.9.4.3	Extensions	271
11.9.4.4	Restrictions	271
11.9.4.5	Modified semantics	272
11.10	Java permissions	272
11.10.1	Permissions for unsigned applications	272
11.10.1.1	java.awt.AWTPermission	272
11.10.1.2	java.net.SocketPermission	272
11.10.1.3	java.util.PropertyPermission	272
11.10.1.4	java.lang.RuntimePermission	272
11.10.1.5	java.io.SerializablePermission	272
11.10.1.6	java.io.FilePermission	273
11.10.1.7	javax.tv.media.MediaSelectPermission	273
11.10.1.8	javax.tv.service.ReadPermission	273
11.10.1.9	javax.tv.service.selection.ServiceContextPermission	273
11.10.1.10	java.util.Locale.setDefault	273
11.10.2	Additional Permissions for signed applications	273
11.10.2.1	java.util.PropertyPermission	273
11.10.2.2	java.io.FilePermission	273

11.10.2.3	org.dvb.net.ca.CAPermission	273
11.10.2.4	org.dvb.application.AppsControlPermission	273
11.10.2.5	org.dvb.net.rc.RCPermission	274
11.10.2.6	org.dvb.net.tuning.TunerPermission	274
11.10.2.7	javax.tv.service.selection.SelectPermission	274
11.10.2.8	org.dvb.user.UserPreferencePermission	274
11.10.2.9	java.net.SocketPermission	274
11.10.2.10	org.dvb.media.DripFeedPermission	274
11.10.2.11	org.dvb.application.storage.ApplicationStoragePermission	274
11.10.2.12	org.dvb.smartcard.SmartCardPermission	274
11.10.2.13	ServiceContextPermission	275
11.11	Content referencing	275
11.11.1	Transport stream	275
11.11.2	Network	275
11.11.3	Bouquet	276
11.11.4	Service	276
11.11.4.1	MPEG/DVB specific service	276
11.11.4.2	Generic Service	277
11.11.5	DVB Event	277
11.11.6	MPEG elementary stream	278
11.11.7	File	278
11.11.8	Directory	279
11.11.9	Drip feed decoder	279
11.11.10	Irrelevant	279
11.11.11	Methods working on many Locator types	279
11.11.12	Support for the HTTP protocol in DVB-J	280
11.12	Stand-alone applications	281
11.12.1	Common behaviour	281
11.12.2	Stored services	281
11.12.2.1	Stored application APIs	281
11.12.2.2	Modified behaviour of MHP 1.0 APIs	281
11.13	Support for DVB-HTML	282
11.13.1	Document object model APIs	282
11.13.1.1	Framework	282
11.13.1.2	DVB defined extensions	282
11.13.1.3	Read Only Access to DOM	282
11.13.2	Embedded Xlets in a DVB-HTML Page	282
11.14	Internet access	283
11.14.1	Internet client control APIs	283
11.14.2	Internet applet support	283
11.14.2.1	HTML tags	283
11.14.2.2	Common MHP features	283
11.14.2.3	Extended or modified common MHP features	283
11.14.2.4	The java.awt package	284
11.14.2.5	Additional features for applet support	284
11.14.2.6	Security	284
12	Security	286
12.1	Introduction	286
12.1.1	Overview of the security framework for applications	286
12.1.2	Overview of return channel security	286
12.2	Authentication of applications	286
12.2.1	Overview of authentication messages	286
12.2.1.1	Hash codes	287
12.2.1.2	Signatures	287
12.2.1.3	Certificates	287
12.2.1.4	Authentication of hierarchical file systems	288
12.3	Message transport	288
12.4	Detail of application authentication messages	289
12.4.1	HashFile	289

12.4.1.1	Description . . . . .	289
12.4.1.2	HashFile location and naming conventions . . . . .	290
12.4.1.3	Digest value computation rules . . . . .	290
12.4.1.4	Special authentication rules . . . . .	290
12.4.2	SignatureFile . . . . .	291
12.4.2.1	Description . . . . .	291
12.4.2.2	SignatureFile location and naming conventions . . . . .	291
12.4.2.3	Supported algorithms . . . . .	292
12.4.2.4	Signature computation rules . . . . .	292
12.4.2.5	Authentication rules . . . . .	292
12.4.3	CertificateFile . . . . .	292
12.4.3.1	Description . . . . .	292
12.4.3.2	ASN.1 encoding . . . . .	293
12.4.3.3	Supported algorithms . . . . .	293
12.4.3.4	Name matching . . . . .	293
12.4.3.5	CertificateFile location and naming conventions . . . . .	293
12.4.3.6	Authentication rules . . . . .	293
12.4.4	Integration . . . . .	293
12.5	Profile of X.509 certificates for authentication of applications . . . . .	294
12.5.1	signatureAlgorithm . . . . .	294
12.5.1.1	MD5 with RSA . . . . .	294
12.5.1.2	SHA-1 with RSA . . . . .	294
12.5.1.3	parameters . . . . .	294
12.5.2	signatureValue . . . . .	294
12.5.3	version . . . . .	294
12.5.4	issuer . . . . .	295
12.5.4.1	minimum requirement . . . . .	295
12.5.4.2	certificate authority responsibility . . . . .	295
12.5.5	validity . . . . .	295
12.5.6	subject . . . . .	295
12.5.7	SubjectPublic Key Info . . . . .	295
12.5.7.1	rsaEncryption . . . . .	295
12.5.7.2	subjectPublicKey . . . . .	295
12.5.8	Unique Identifiers . . . . .	296
12.5.9	Extensions . . . . .	296
12.6	Security policy for applications . . . . .	297
12.6.1	General principles . . . . .	297
12.6.2	Permission request file . . . . .	298
12.6.2.1	File encoding . . . . .	298
12.6.2.1.1	XML . . . . .	298
12.6.2.1.2	MHP 1.0 . . . . .	298
12.6.2.1.3	MHP 1.1 . . . . .	299
12.6.2.1.4	Number representation . . . . .	301
12.6.2.2	File integrity . . . . .	301
12.6.2.3	Example . . . . .	301
12.6.2.4	Permission request file name and location . . . . .	302
12.6.2.5	Permission request file . . . . .	302
12.6.2.5.1	Minimum permissions . . . . .	302
12.6.2.5.2	Syntax and semantics . . . . .	302
12.6.2.6	Credentials . . . . .	303
12.6.2.7	File Access . . . . .	305
12.6.2.7.1	Unsigned applications . . . . .	305
12.6.2.7.2	Default policy for signed applications . . . . .	305
12.6.2.7.3	Permission request syntax . . . . .	306
12.6.2.8	CA API . . . . .	306
12.6.2.8.1	Unsigned applications . . . . .	306
12.6.2.8.2	Signed applications . . . . .	306
12.6.2.8.3	Conditional Access Permission syntax . . . . .	307
12.6.2.9	Application lifecycle control policy . . . . .	307

12.6.2.9.1	Unsigned applications	307
12.6.2.9.2	Default policy for Signed applications	307
12.6.2.9.3	Syntax	307
12.6.2.10	Return channel access policy	307
12.6.2.10.1	Unsigned applications	307
12.6.2.10.2	Signed applications	307
12.6.2.10.3	Return channel permission syntax	308
12.6.2.11	Tuning access policy	308
12.6.2.11.1	Unsigned applications	308
12.6.2.11.2	Signed applications	308
12.6.2.11.3	Tuner Permission syntax	308
12.6.2.12	Service selection policy	308
12.6.2.12.1	Unsigned applications	308
12.6.2.12.2	Signed applications	308
12.6.2.12.3	Service Selection Permission	309
12.6.2.13	Media API access policy	309
12.6.2.14	Inter-application communication policy	309
12.6.2.14.1	Unsigned applications	309
12.6.2.14.2	Signed applications	309
12.6.2.15	User Setting and Preferences access policy	309
12.6.2.15.1	Unsigned applications	309
12.6.2.15.2	Signed applications	309
12.6.2.15.3	Permission syntax	309
12.6.2.16	Network permissions	310
12.6.2.16.1	Unsigned applications	310
12.6.2.16.2	Signed applications	310
12.6.2.16.3	Permission syntax	310
12.6.2.17	Dripfeed permissions	310
12.6.2.17.1	Unsigned applications	310
12.6.2.17.2	Default policy for signed applications	310
12.6.2.17.3	Permission request syntax	310
12.6.2.18	Privileged Runtime Code Extension Permission	310
12.6.2.18.1	Unsigned Applications	310
12.6.2.18.2	Signed applications with no permission request file	310
12.6.2.18.3	Signed applications with a permission request file	311
12.6.2.18.4	Permission request syntax	311
12.6.2.19	Application storage	311
12.6.2.19.1	Unsigned applications	311
12.6.2.19.2	Default policy for signed applications	311
12.6.2.19.3	Permission request syntax	311
12.6.2.20	Non-CA smart card access	311
12.6.2.20.1	Unsigned applications	311
12.6.2.20.2	Default policy for signed applications	311
12.6.2.20.3	Permission request syntax	312
12.7	Example of creating an application that can be authenticated	312
12.7.1	Scenario Example	312
12.7.2	Hashes and signature computations	313
12.7.2.1	Computation of the hashes of the root/Xlet1/classes/subclasses directory	313
12.7.2.2	Computation of the hashes of the of root/Xlet1/classes directory	313
12.7.2.3	Computation of the hashes of the of root/Xlet1 directory	314
12.7.2.4	Computation of the signature	314
12.8	Procedures for application certificates and signatures	314
12.9	Certificate management	315
12.9.1	Certificate Revocation Lists	315
12.9.1.1	Introduction (informative)	315
12.9.1.2	Distribution of CRLs (informative)	315
12.9.1.2.1	Distribution via return channel	315
12.9.1.2.2	Distribution via MPEG stream	315
12.9.1.3	CRL retention	315

12.9.1.3.1	Requirement . . . . .	315
12.9.1.3.2	Storage requirement . . . . .	315
12.9.1.3.3	Storage management . . . . .	315
12.9.1.4	CRL file location and naming convention . . . . .	316
12.9.1.5	Operational model . . . . .	316
12.9.1.6	Examples . . . . .	316
12.9.1.6.1	Revocation of a broadcaster's certificate . . . . .	316
12.9.1.6.2	Revocation of a CA's certificate. . . . .	316
12.9.1.7	CRL format . . . . .	317
12.9.1.8	Profile of CRL . . . . .	317
12.9.1.9	CRL Processing . . . . .	318
12.9.2	Root certificate management . . . . .	318
12.9.2.1	Introduction . . . . .	318
12.9.2.2	Security of RCMM . . . . .	318
12.9.2.3	Format of RCMM . . . . .	319
12.9.2.4	Distribution of RCMM . . . . .	319
12.9.2.5	RCMM Processing. . . . .	319
12.9.2.6	Example: Renewal of a root certificate . . . . .	320
12.10	Security on the return channel. . . . .	320
12.10.1	MHP functionality . . . . .	320
12.10.2	TLS cipher suites . . . . .	321
12.10.3	The EDE112 cipher . . . . .	321
12.10.4	Downloading of certificates for TLS. . . . .	321
12.10.4.1	Introduction . . . . .	321
12.10.4.2	Usage of certificate in TLS . . . . .	322
12.10.4.2.1	When certificates are delivered with the application . . . . .	322
12.10.4.2.2	When no certificates are provided . . . . .	322
12.11	The internet profile of X.509 (informative) . . . . .	322
12.11.1	Main part of the certificate . . . . .	323
12.11.1.1	Certificate. . . . .	323
12.11.1.2	signatureAlgorithm . . . . .	323
12.11.1.3	signatureValue . . . . .	323
12.11.1.4	tbsCertificate . . . . .	323
12.11.1.5	version . . . . .	324
12.11.1.6	serialNumber . . . . .	324
12.11.1.7	signature. . . . .	324
12.11.1.8	issuer . . . . .	324
12.11.1.9	validity . . . . .	325
12.11.1.9.1	UTCTime . . . . .	325
12.11.1.9.2	GeneralizedTime . . . . .	325
12.11.1.10	subject . . . . .	325
12.11.1.10.1	issuerUniqueID . . . . .	325
12.11.1.10.2	subjectUniqueID . . . . .	326
12.11.1.11	SubjectPublic Key Info . . . . .	326
12.11.1.12	Unique Identifiers . . . . .	326
12.11.1.13	Extensions . . . . .	327
12.11.2	Standard certificate extensions . . . . .	327
12.11.2.1	Authority key identifier . . . . .	327
12.11.2.2	Subject key identifier . . . . .	327
12.11.2.3	Key usage. . . . .	327
12.11.2.4	Private key usage period . . . . .	327
12.11.2.5	Certificate policies . . . . .	328
12.11.2.6	Policy mappings. . . . .	328
12.11.2.7	Subject Alternative Name . . . . .	328
12.11.2.8	Issuer Alternative Name . . . . .	329
12.11.2.9	Subject Directory attributes . . . . .	329
12.11.2.10	Basic Constraints . . . . .	329
12.11.2.11	Name Constraints. . . . .	329
12.11.2.12	Policy Constraints . . . . .	329

12.11.2.13	Extended key usage field . . . . .	330
12.11.2.14	CRL Distribution points . . . . .	330
12.12	MHP certification procedures . . . . .	330
12.13	Plug-ins . . . . .	331
12.14	Applications loaded from an interaction channel . . . . .	331
12.14.1	Permission for application loading from the return channel . . . . .	331
12.15	Stored applications . . . . .	331
12.16	Inner applications and content embedded within other applications . . . . .	332
13	Graphics reference model . . . . .	333
13.1	Introduction . . . . .	333
13.1.1	Interapplication interaction . . . . .	333
13.2	General Issues . . . . .	334
13.2.1	Coordinate Spaces . . . . .	334
13.2.1.1	Normalised screen space . . . . .	334
13.2.1.2	User space . . . . .	335
13.2.1.3	Pixel Aspect Ratio . . . . .	337
13.2.1.4	Video space . . . . .	338
13.3	Graphics . . . . .	338
13.3.1	Modelling of the MHP display stack composition . . . . .	338
13.3.2	AWT Reference Model in the MHP . . . . .	340
13.3.3	HAVi devices and AWT components . . . . .	341
13.3.3.1	Video and graphics pixel aligned . . . . .	343
13.3.3.2	Zero graphics impact . . . . .	343
13.3.4	Composition . . . . .	343
13.3.4.1	AWT paint rule . . . . .	343
13.3.5	Composition Rules . . . . .	344
13.3.5.1	Components generally . . . . .	344
13.3.6	Extensions to the AWT graphics capabilities . . . . .	345
13.3.6.1	Graphics Objects in the MHP . . . . .	345
13.3.6.2	Buffered Image . . . . .	345
13.3.6.3	DVBColour . . . . .	345
13.3.6.3.1	Modified packed colour representation . . . . .	345
13.4	Video . . . . .	346
13.4.1	Component-based players and background players . . . . .	346
13.4.2	Modelling MPEG decoding and presentation pipeline . . . . .	346
13.4.3	Coordinate Spaces . . . . .	347
13.4.4	Video components . . . . .	348
13.5	Subtitles . . . . .	349
13.5.1	Language and presentation setting . . . . .	349
13.5.2	Relation to graphics . . . . .	349
13.5.3	Coordinate Spaces . . . . .	349
13.6	Approximations . . . . .	350
13.6.1	Approximations in composition . . . . .	350
13.6.1.1	Implementation of modes . . . . .	350
13.6.1.1.1	Graphics directly over video . . . . .	350
13.6.1.1.2	Graphics over other graphics . . . . .	350
13.6.1.2	Approximation of alpha . . . . .	351
13.6.1.3	Approximation of colour . . . . .	352
14	System integration aspects . . . . .	353
14.1	Namespace mapping (DVB Locator) . . . . .	353
14.1.1	dvb_entity = dvb_service . . . . .	353
14.1.2	dvb_entity = dvb_service_component . . . . .	354
14.1.3	dvb_hier_part = dvb_abs_path . . . . .	354
14.1.4	dvb_abs_path . . . . .	354
14.2	Reserved names . . . . .	354
14.3	XML notation . . . . .	355
14.4	Network signalling . . . . .	357
14.5	Text encoding of application identifiers . . . . .	357

14.6	Reserved names for persistent storage . . . . .	357
14.7	Files and file names. . . . .	358
14.8	Locators and content referencing . . . . .	358
14.9	Service identification . . . . .	359
14.9.1	Syntax of the textual service identifier . . . . .	359
14.9.2	Handling of the textual service identifiers within the MHP terminal . . . . .	360
15	Detailed platform profile definitions . . . . .	361
15.1	PNG - restrictions . . . . .	363
15.1.1	PNG Aspect ratios . . . . .	363
15.2	Minimum media formats supported by DVB-J APIs . . . . .	364
15.3	JPEG - restrictions. . . . .	364
15.4	Locale support. . . . .	364
16	Registry of Constants. . . . .	365
16.1	System constants. . . . .	365
16.2	DVB-J constants . . . . .	365
16.2.1	Public and Protected final static primitive fields from DVB packages. . . . .	365
16.2.2	Public and Protected final static primitive fields from standard Java packages . . . . .	368
17	Internet access clients. . . . .	369
17.1	Referencing DVB services and content within WWW content . . . . .	369
17.2	Minimum requirements for internet clients. . . . .	369
17.3	Internet streamed media . . . . .	369
<b>Annex A (normative): External references; errata, clarifications and exemptions. . . . .</b>		<b>370</b>
A.1	<a href="#">JAE 1.1.8 API [31]</a> . . . . .	370
A.1.1	java.lang.ThreadGroup.getParent() . . . . .	370
A.1.2	java.net.URLConnection.setFileNameMap . . . . .	370
A.1.3	java.util.Locale.setDefault . . . . .	370
A.1.4	java.lang.Class . . . . .	370
A.1.5	java.awt.Font . . . . .	370
A.2	<a href="#">Java Language Spec [32]</a> . . . . .	370
A.2.1	java.lang.ThreadGroup.getParent() . . . . .	370
A.2.2	java.lang.Runtime.runFinalizersOnExit() . . . . .	370
A.2.3	java.lang.System.runFinalizersOnExit() . . . . .	370
A.3	<a href="#">Java Media Player Specification [33]</a> . . . . .	371
A.3.1	javax.media.protocol.URLDataSource.sources . . . . .	371
A.4	<a href="#">Java VM [34]</a> . . . . .	371
A.5	<a href="#">Java TV [51]</a> . . . . .	371
A.5.1	javax.tv.service.selection . . . . .	371
A.5.1.1	PresentationTerminatedEvent . . . . .	371
A.5.1.2	ServiceContext.select( Locator [] ) . . . . .	372
A.5.2	javax.tv.util.TVTimer . . . . .	372
A.5.2.1	scheduleTimerSpec(TVTimerSpec) . . . . .	372
A.5.3	javax.tv.util.TVTimerSpec . . . . .	372
A.5.3.1	setAbsoluteTime(long) . . . . .	372
A.5.3.2	setTime(long). . . . .	372
A.5.4	javax.tv.xlet.Xlet . . . . .	372
A.5.4.1	Xlet state descriptions . . . . .	372
A.5.4.2	Own service context. . . . .	372
A.6	<a href="#">DAVIC 1.4.1p9 [3]</a> . . . . .	373
A.6.1	org.davic.mpeg.NotAuthorizedException . . . . .	373
A.6.2	org.davic.mpeg.dvb . . . . .	376
A.6.2.1	DvbElementaryStream . . . . .	376
A.6.3	org.davic.mpeg.sections . . . . .	376
A.6.3.1	RingSectionFilter . . . . .	376
A.6.3.2	Section . . . . .	376
A.6.3.2.1	clone() . . . . .	376
A.6.3.2.2	Started Section Filters . . . . .	376
A.6.3.2.3	getData(). . . . .	376



A.6.3.2.4	getFullStatus() .....	376
A.6.3.3	SectionFilter .....	376
A.6.3.3.1	Cross reference error .....	376
A.6.3.3.2	startFiltering(java.lang.Object, int, int) .....	377
A.6.3.3.3	startFiltering(java.lang.Object, int, int, int, byte[], byte[]) .....	377
A.6.3.3.4	startFiltering (appData, pid, tableId) exceptions .....	377
A.6.3.4	SectionFilterGroup .....	377
A.6.3.4.1	NotAuthorizedException .....	377
A.6.3.4.2	numberOfFilters .....	377
A.6.3.4.3	sectionSize .....	377
A.6.3.4.4	ringSize .....	377
A.6.3.5	TimeoutEvent .....	377
A.6.4	org.davic.media .....	378
A.6.4.1	FreezeControl.resume() .....	378
A.6.4.2	MediaTimePositionChangedEvent .....	378
A.6.4.3	NotAuthorizedMediaException .....	378
A.6.5	org.davic.net .....	379
A.6.5.1	InvalidLocatorException .....	379
A.6.5.2	Locator .....	379
A.6.5.2.1	Locator() .....	379
A.6.5.2.2	toExternalForm() .....	379
A.6.5.3	tuning .....	380
A.6.5.3.1	NetworkInterfaceController .....	380
A.6.5.4	ca .....	380
A.6.5.4.1	CAMessage .....	380
A.6.5.4.2	CAModule .....	381
A.6.5.4.3	CAModuleManager .....	383
A.6.5.4.4	NoFreeCapacityException .....	383
A.6.5.4.5	MMIOObject .....	384
A.6.5.4.6	DescramblerProxy .....	384
A.6.5.4.7	StartMMIEvent(MMIOObject, int, java.lang.Object) .....	385
A.6.5.5	dvb.DvbLocator(int onid, int tsid, int serviceid, int eventid, int componenttags[], String filePath) .....	385
A.6.5.6	dvb.DvbLocator .....	385
A.7	CSS 2 [87] .....	386
<b>Annex B (normative): Object carousel .....</b>		<b>388</b>
B.1	Introduction .....	388
B.1.1	Key to notation .....	388
B.2	Object Carousel Profile .....	388
B.2.1	DSM-CC Sections .....	388
B.2.1.1	Sections per TS packet .....	389
B.2.2	Data Carousel .....	389
B.2.2.1	General .....	389
B.2.2.2	DownloadInfoIndication .....	389
B.2.2.3	DownloadServerInitiate .....	389
B.2.2.4	ModuleInfo .....	390
B.2.2.4.1	Label descriptor .....	390
B.2.2.4.2	Caching priority descriptor .....	391
B.2.2.5	ServiceGatewayInfo .....	392
B.2.2.6	Download Cancel .....	392
B.2.3	The Object Carousel .....	393
B.2.3.1	BIOP Generic Object Message .....	393
B.2.3.2	CORBA strings .....	393
B.2.3.3	BIOP FileMessage .....	393
B.2.3.4	Content type descriptor .....	394
B.2.3.5	BIOP DirectoryMessage .....	395
B.2.3.6	BIOP ServiceGateway message .....	397
B.2.3.7	BIOP Interoperable Object References .....	397

B.2.3.7.1	BIOPProfileBody . . . . .	398
B.2.3.7.2	LiteOptionsProfileBody . . . . .	399
B.2.3.8	BIOP StreamMessage . . . . .	401
B.2.3.9	BIOP StreamEventMessage . . . . .	403
B.2.4	Stream Events . . . . .	404
B.2.4.1	Stream & StreamEvent messages . . . . .	404
B.2.4.1.1	Association with time bases . . . . .	404
B.2.4.1.2	Event names and event ids . . . . .	405
B.2.4.1.3	Stream event life time . . . . .	405
B.2.4.2	Stream Descriptors . . . . .	405
B.2.4.2.1	NPT Reference descriptor . . . . .	405
B.2.4.2.2	Stream event descriptor . . . . .	406
B.2.4.2.3	Unused descriptors . . . . .	406
B.2.4.2.4	Clarification of number encoding . . . . .	406
B.2.4.3	DSM-CC Sections carrying Stream Descriptors . . . . .	407
B.2.4.3.1	Section version number . . . . .	407
B.2.4.3.2	Single firing of "do it now" events . . . . .	407
B.2.4.3.3	Section number . . . . .	407
B.2.4.3.4	DSM-CC sections for DSMCC_descriptor_list() . . . . .	407
B.2.4.3.5	Encoding of table id extension . . . . .	407
B.2.4.4	Timebases . . . . .	407
B.2.4.5	Monitoring stream events . . . . .	408
B.2.4.5.1	NPT reference monitoring . . . . .	408
B.2.4.5.2	Timebase stimulated event monitoring . . . . .	408
B.2.4.5.3	"do it now" events . . . . .	408
B.2.4.5.4	scheduled events . . . . .	409
B.2.4.5.5	number of NPT components . . . . .	409
B.2.5	Assignment and use of transactionId values . . . . .	409
B.2.6	Mapping of objects to data carousel modules . . . . .	410
B.2.7	Compression of modules . . . . .	410
B.2.8	Mounting an Object Carousel . . . . .	411
B.2.8.1	carousel_id_descriptor . . . . .	412
B.2.8.2	DVB-J mounting of an object carousel . . . . .	413
B.2.9	Unavailability of a carousel . . . . .	413
B.3	AssociationTag Mapping . . . . .	414
B.3.1	Decision algorithm for association tag mapping . . . . .	414
B.3.1.1	TapUse is <b>not</b> BIOP_PROGRAM_USE . . . . .	414
B.3.1.2	TapUse <b>is</b> BIOP_PROGRAM_USE . . . . .	415
B.3.2	DSM-CC association_tags to DVB component_tags . . . . .	415
B.3.3	deferred_association_tag_descriptor . . . . .	415
B.4	Example of an Object Carousel (informative) . . . . .	416
B.5	Caching . . . . .	417
B.5.1	Determining file version . . . . .	417
B.5.2	Transparency levels of caching . . . . .	417
B.5.2.1	Transparent caching . . . . .	417
B.5.2.1.1	Active caching . . . . .	418
B.5.2.1.2	Passive caching . . . . .	418
B.5.2.1.3	DII repetition rate . . . . .	418
B.5.2.2	Semi-transparent caching . . . . .	418
B.5.2.2.1	Implications for the terminal (informative) . . . . .	418
B.5.2.3	Static caching . . . . .	419
B.5.2.3.1	Implications for the broadcaster (informative) . . . . .	419
B.5.2.3.2	Implications for the terminal (informative) . . . . .	419
<b>Annex C (informative):</b>	<b>References . . . . .</b>	<b>420</b>
<b>Annex D (normative):</b>	<b>Text presentation . . . . .</b>	<b>421</b>
D.1	Scope . . . . .	421
D.2	Fonts . . . . .	421
D.2.1	Embedded fonts . . . . .	421

D.2.2	Downloaded fonts	421
D.2.2.1	Font technology	421
D.2.2.2	Font index files	422
D.2.2.2.1	Format of file	422
D.2.2.2.2	Element semantics	422
D.2.2.2.3	Example	423
D.2.2.3	Name and location of font index files	423
D.2.2.3.1	General	423
D.2.2.3.2	Name of file	423
D.2.2.3.3	Location	423
D.2.2.4	Specification of fonts at run time	423
D.2.2.4.1	DVB-J	423
D.3	Text rendering	424
D.3.1	Philosophy	424
D.3.2	Low and high level rendering	424
D.3.2.1	Low level rendering	424
D.3.2.2	High level rendering	424
D.3.3	Font Definition	424
D.3.3.1	"Physical" font data	425
D.3.4	Converting font metrics to display pixels	425
D.3.4.1	Vertical resolution	426
D.3.4.2	Horizontal resolution	426
D.3.5	Rendering within limits	426
D.3.5.1	Vertical limits	427
D.3.5.2	Horizontal limits	428
D.3.6	"logical" text width rules	428
D.3.6.1	Computing "logical" text width	428
D.3.6.1.1	Font sizes	429
D.3.6.1.2	Character widths	429
D.3.6.1.3	Kerning	429
D.3.6.1.4	Tracking	429
D.3.6.2	Logical text width	429
D.3.7	Line breaking	430
D.3.7.1	Truncation of text	430
D.3.8	Tabulation	430
D.3.9	Placing runs of characters & words	431
D.3.10	Control of text flow	432
D.4	Text mark-up	432
D.4.1	White Space Characters	432
D.4.2	Marker characters	432
D.4.3	Non-printing characters	432
D.4.4	Format Control Mark-up	433
D.4.5	Future compatibility	433
<b>Annex E (normative): Character set</b>		<b>434</b>
E.1	Basic Euro Latin character set	434
<b>Annex F (informative): Authoring &amp; Implementation Guidelines</b>		<b>439</b>
F.1	Authoring Guidelines	439
F.2	Implementation Guidelines	439
F.3	Authoring guidelines for DVB-J	439
F.4	Authoring guidelines for DVB HTML	439
F.4.1	CSS2 Authoring guidelines	439
F.4.1.1	Selectors	439
F.4.1.2	Properties	439
F.4.1.2.1	Generalities	439
F.4.1.2.2	Visual Formatting Model	439
F.4.1.2.3	Colours and Background	440
F.4.1.2.4	Visual Effects	440
F.4.1.2.5	Fonts	440

F.4.1.2.6	Text .....	440
F.4.1.2.7	User Interface .....	441
<b>Annex G (normative):</b>	<b>Minimum Platform Capabilities .....</b>	<b>442</b>
G.1	Graphics .....	442
G.1.1	Device capabilities .....	442
G.1.2	Video presentation capabilities .....	442
G.1.3	Image processing capabilities .....	443
G.1.4	Alpha capabilities .....	443
G.1.5	Colour capabilities .....	443
G.1.6	MPEG I frame and Video drips .....	444
G.2	Audio .....	444
G.3	Video .....	445
G.4	Resident fonts and text rendering .....	445
G.4.1	The built-in font .....	445
G.4.2	Presentation to DVB-J .....	445
G.4.3	Text directions .....	445
G.5	Input events .....	446
G.6	Memory .....	446
G.7	Other resources .....	447
<b>Annex H (normative):</b>	<b>DVB-J fundamental classes .....</b>	<b>449</b>
	DVBClassLoader .....	451
<b>Annex I (normative):</b>	<b>DVB-J event API .....</b>	<b>453</b>
I.1	Overview .....	453
I.2	The resource management .....	454
I.3	The Event Repository .....	454
I.3.1	Example .....	454
	EventManager .....	457
	OverallRepository .....	460
	RepositoryDescriptor .....	461
	UserEvent .....	462
	UserEventListener .....	465
	UserEventRepository .....	466
<b>Annex J (normative):</b>	<b>DVB-J persistent storage API .....</b>	<b>469</b>
	FileAccessPermissions .....	471
	FileAttributes .....	473
<b>Annex K (informative):</b>	<b>Extensions .....</b>	<b>476</b>
<b>Annex L (normative):</b>	<b>User Settings and Preferences API .....</b>	<b>477</b>
	Facility .....	479
	GeneralPreference .....	480
	Preference .....	481
	UnsupportedPreferenceException .....	484
	UserPreferenceChangeEvent .....	485
	UserPreferenceChangeListener .....	486
	UserPreferenceManager .....	487
	UserPreferencePermission .....	489
<b>Annex M (normative):</b>	<b>SI Access API .....</b>	<b>490</b>
	Descriptor .....	494
	DescriptorTag .....	496
	PMTElementaryStream .....	500
	PMTService .....	502
	PMTStreamType .....	504
	SIBouquet .....	505
	SIDatabase .....	509

SIEvent	525
SIException	529
SIIllegalArgumentException	530
SIInformation	531
SIInvalidPeriodException	535
SIIterator	536
SILackOfResourcesEvent	537
SIMonitoringEvent	538
SIMonitoringListener	541
SIMonitoringType	542
SINetwork	543
SINotInCacheEvent	547
SIOBJECTNotInTableEvent	548
SIRequest	549
SIRequestCancelledEvent	550
SIRetrievalEvent	551
SIRetrievalListener	553
SIRunningStatus	554
SIService	555
SIServiceType	561
SISuccessfulRetrieveEvent	563
SITableNotFoundEvent	564
SITableUpdatedEvent	565
SITime	566
SITransportStream	567
SITransportStreamBAT	569
SITransportStreamDescription	570
SITransportStreamNIT	571
SIUtil	572
TextualServiceIdentifierQuery	573
<b>Annex N (normative): Streamed Media API Extensions</b>	<b>574</b>
ActiveFormatDescriptionChangedEvent	577
AspectRatioChangedEvent	578
BackgroundVideoPresentationControl	579
CAException	580
CAStopEvent	581
DFCChangedEvent	583
DripFeedDataSource	584
DripFeedPermission	587
NoComponentSelectedEvent	588
PresentationChangedEvent	590
ServiceRemovedEvent	592
StopByResourceLossEvent	594
SubtitleAvailableEvent	596
SubtitleListener	597
SubtitleNotAvailableEvent	598
SubtitleNotSelectedEvent	599
SubtitleSelectedEvent	600
SubtitlingEventControl	601
VideoFormatControl	602
VideoFormatEvent	607
VideoFormatListener	608
VideoPresentationControl	609
VideoTransformation	614
<b>Annex O (normative): Integration of the JavaTV SI API and DVB SI</b>	<b>617</b>
O.1 Introduction	617
O.2 Mapping of the JavaTV SI API to DVB SI	617
O.2.1 javax.tv.service.Service	617

O.2.1.1	getName . . . . .	617
O.2.1.2	getServiceType . . . . .	617
O.2.2	javax.tv.service.ServiceComponent . . . . .	617
O.2.2.1	getComponentName . . . . .	617
O.2.2.2	getAssociatedLanguage . . . . .	617
O.2.2.3	getStreamType . . . . .	617
O.2.3	javax.tv.service.ServiceType . . . . .	618
O.2.4	javax.tv.service.StreamType . . . . .	618
O.2.5	javax.tv.service.SIElement . . . . .	618
O.2.5.1	getServiceInformationType . . . . .	618
O.2.6	javax.tv.service.SIManager . . . . .	619
O.2.6.1	getSupportedDimensions . . . . .	619
O.2.6.2	getRatingDimension . . . . .	619
O.2.6.3	retrieveSIElement . . . . .	619
O.2.6.4	getTransports . . . . .	619
O.2.6.5	filterServices . . . . .	619
O.2.7	javax.tv.service.navigation.SIElementFilter . . . . .	619
O.2.8	javax.tv.service.navigation.ServiceDetails . . . . .	619
O.2.8.1	getLongName . . . . .	619
O.2.8.2	getServiceType . . . . .	619
O.2.8.3	retrieveServiceDescription . . . . .	619
O.2.8.4	retrieveComponents . . . . .	619
O.2.9	javax.tv.service.navigation.CAIdentification . . . . .	620
O.2.9.1	getCASystemIds . . . . .	620
O.2.9.2	isFree . . . . .	620
O.2.10	javax.tv.service.navigation.RatingDimension . . . . .	620
O.2.10.1	getDimensionName . . . . .	620
O.2.10.2	getNumberOfLevels . . . . .	620
O.2.10.3	getRatingLevelDescription . . . . .	620
O.2.11	javax.tv.service.navigation.ServiceProviderInformation . . . . .	620
O.2.11.1	getProviderName . . . . .	620
O.2.12	javax.tv.service.transport.Transport . . . . .	620
O.2.13	javax.tv.service.transport.Bouquet . . . . .	620
O.2.13.1	getBouquetID . . . . .	621
O.2.13.2	getName . . . . .	621
O.2.13.3	getLocator . . . . .	621
O.2.14	javax.tv.service.transport.Network . . . . .	621
O.2.14.1	getNetworkID . . . . .	621
O.2.14.2	getName . . . . .	621
O.2.14.3	getLocator . . . . .	621
O.2.15	javax.tv.service.transport.TransportStream . . . . .	621
O.2.15.1	getTransportStreamID . . . . .	621
O.2.15.2	getDescription . . . . .	621
O.2.16	javax.tv.service.guide.ProgramEvent . . . . .	621
O.2.16.1	getDuration . . . . .	621
O.2.16.2	getStartTime . . . . .	621
O.2.16.3	getEndTime . . . . .	621
O.2.16.4	getName . . . . .	622
O.2.16.5	retrieveDescription . . . . .	622
O.2.16.6	getRating . . . . .	622
O.2.17	javax.tv.service.guide.ContentRatingAdvisory . . . . .	622
O.2.17.1	getDimensionNames . . . . .	622
O.2.17.2	getRatingLevel . . . . .	622
O.2.17.3	getRatingText . . . . .	622
O.2.17.4	getDisplayText . . . . .	622
O.3	Integration of the JavaTV SI API and the DVB SI API . . . . .	623
<b>Annex P (normative): Broadcast Transport Protocol Access . . . . .</b>		<b>624</b>
	AsynchronousLoadingEvent . . . . .	628

AsynchronousLoadingEventListener . . . . .	629
DSMCCException . . . . .	630
DSMCCObject . . . . .	631
DSMCCStream . . . . .	637
DSMCCStreamEvent . . . . .	641
IllegalObjectTypeException . . . . .	644
InsufficientResourcesException . . . . .	645
InvalidAddressException . . . . .	646
InvalidFormatEvent . . . . .	647
InvalidFormatException . . . . .	648
InvalidPathnameEvent . . . . .	649
InvalidPathNameException . . . . .	650
LoadingAbortedEvent . . . . .	651
MPEGDeliveryErrorEvent . . . . .	652
MPEGDeliveryException . . . . .	653
NotEntitledEvent . . . . .	654
NotEntitledException . . . . .	655
NothingToAbortException . . . . .	656
NotLoadedException . . . . .	657
NPTDiscontinuityEvent . . . . .	658
NPTListener . . . . .	660
NPTPresentEvent . . . . .	661
NPTRate . . . . .	662
NPTRateChangeEvent . . . . .	663
NPTRemovedEvent . . . . .	664
NPTStatusEvent . . . . .	665
ObjectChangeEvent . . . . .	666
ObjectChangeEventListener . . . . .	668
ServerDeliveryErrorEvent . . . . .	669
ServerDeliveryException . . . . .	670
ServiceDomain . . . . .	671
ServiceXFRErrorEvent . . . . .	675
ServiceXFRException . . . . .	676
ServiceXFRReference . . . . .	678
StreamEvent . . . . .	680
StreamEventListener . . . . .	682
SuccessEvent . . . . .	683
UnknownEventException . . . . .	684
<b>Annex Q (normative): Datagram Socket Buffer Control . . . . .</b>	<b>685</b>
DatagramSocketBufferControl . . . . .	687
<b>Annex R (normative): DVB-J Return Channel Connection Management API . . . . .</b>	<b>688</b>
ConnectionEstablishedEvent . . . . .	690
ConnectionFailedEvent . . . . .	691
ConnectionListener . . . . .	692
ConnectionParameters . . . . .	693
ConnectionRCEvent . . . . .	695
ConnectionRCInterface . . . . .	696
ConnectionTerminatedEvent . . . . .	700
IncompleteTargetException . . . . .	701
PermissionDeniedException . . . . .	702
RCInterface . . . . .	703
RCInterfaceManager . . . . .	705
RCInterfaceReleasedEvent . . . . .	707
RCInterfaceReservedEvent . . . . .	708
RCPermission . . . . .	709
<b>Annex S (normative): Application Listing and Launching . . . . .</b>	<b>711</b>
AppAttributes . . . . .	713

AppIcon	718
AppID	719
AppProxy	721
AppsControlPermission	725
AppsDatabase	727
AppsDatabaseEvent	731
AppsDatabaseEventListener	733
AppsDatabaseFilter	735
AppStateChangeEvent	736
AppStateChangeListener	738
CurrentServiceFilter	739
DVBHTMLProxy	740
DVBJProxy	742
IllegalProfileParameterException	744
LanguageNotAvailableException	745
<b>Annex T (normative): Permissions</b>	<b>746</b>
CAPermission	748
DvbNetworkInterfaceSIUtil	751
TunerPermission	752
<b>Annex U (normative): Extended graphics APIs</b>	<b>753</b>
DVBAAlphaComposite	755
DVBBufferedImage	760
DVBColor	767
DVBBGraphics	771
DVBRasterFormatException	775
DVBBTextLayoutManager	776
FontFactory	783
FontFormatException	785
FontNotAvailableException	786
TestOpacity	787
TextOverflowListener	788
UnsupportedDrawingOperationException	789
<b>Annex V (normative): HAVi Level 2 User Interface</b>	<b>790</b>
V.1 Register of Constants	790
HActionable	800
HActionInputPreferred	804
HAdjustableLook	805
HAdjustmentInputPreferred	808
HAdjustmentValue	810
HAnimateEffect	814
HAnimateLook	818
HAnimation	823
HBackgroundConfigTemplate	830
HBackgroundConfiguration	833
HBackgroundDevice	835
HBackgroundImage	839
HChangeData	842
HComponent	843
HComponentOrdering	846
HConfigurationException	850
HContainer	852
HDefaultTextLayoutManager	859
HEmulatedGraphicsConfiguration	861
HEmulatedGraphicsDevice	863
HEventMulticaster	865
HFlatEffectMatte	877
HFlatMatte	882



HFontCapabilities	884
HGraphicButton	895
HGraphicLook	904
HGraphicsConfigTemplate	909
HGraphicsConfiguration	913
HGraphicsDevice	918
HIcon	922
HImageEffectMatte	928
HImageHints	933
HImageMatte	935
HInvalidLookException	937
HItemValue	938
HKeyboardInputPreferred	941
HListElement	944
HListGroup	946
HListGroupLook	961
HLook	967
HMatte	973
HMatteException	974
HMatteLayer	976
HMultilineEntry	977
HMultilineEntryLook	981
HNavigable	987
HNavigationInputPreferred	992
HNoInputPreferred	994
HOrientable	995
HPermissionDeniedException	997
HRange	999
HRangeLook	1006
HRangeValue	1011
HScene	1020
HSceneFactory	1033
HSceneTemplate	1037
HScreen	1041
HScreenConfigTemplate	1046
HScreenConfiguration	1052
HScreenDevice	1054
HScreenDimension	1058
HScreenPoint	1060
HScreenRectangle	1062
HSelectionInputPreferred	1065
HSinglelineEntry	1067
HSinglelineEntryLook	1079
HSound	1084
HState	1087
HStaticAnimation	1091
HStaticIcon	1097
HStaticRange	1100
HStaticText	1106
HStillImageBackgroundConfiguration	1110
HSwitchable	1113
HText	1116
HTextButton	1123
HTextLayoutManager	1132
HTextLook	1135
HTextValue	1140
HToggleButton	1142
HToggleGroup	1153
HUIException	1157

HVersion	1159
HVideoComponent	1161
HVideoConfigTemplate	1163
HVideoConfiguration	1166
HVideoDevice	1168
HVisible	1173
HActionEvent	1196
HActionListener	1198
HAdjustmentEvent	1199
HAdjustmentListener	1202
HBackgroundImageEvent	1203
HBackgroundImageListener	1206
HEventRepresentation	1207
HFocusEvent	1213
HFocusListener	1216
HItemEvent	1217
HItemListener	1220
HKeyCapabilities	1221
HKeyEvent	1223
HKeyListener	1225
HMouseCapabilities	1226
HRcCapabilities	1227
HRcEvent	1229
HScreenConfigurationEvent	1238
HScreenConfigurationListener	1239
HScreenDeviceReleasedEvent	1240
HScreenDeviceReservedEvent	1241
HScreenLocationModifiedEvent	1242
HScreenLocationModifiedListener	1244
HTextEvent	1245
HTextListener	1249
<b>Annex W (normative): DVB-J examples</b>	<b>1250</b>
W.1 DVB-J Application lifecycle implementation example	1250
W.2 Example of exporting an object for inter-application communication	1251
<b>Annex X (normative): Test support</b>	<b>1252</b>
DVBTest	1255
<b>Annex Y (normative): Inter-application and Inter-Xlet communication API</b>	<b>1261</b>
IxcRegistry	1263
<b>Annex Z (informative): Services, Service Contexts and Applications in an MHP Environment</b>	<b>1267</b>
Z.1 Introduction	1267
Z.2 Basic concepts	1267
Z.3 Presenting a service in MHP	1267
Z.3.1 Presenting the media components of a service	1267
Z.3.2 Presenting the application components of a service	1267
Z.4 Multiple service contexts in an MHP platform	1268
Z.5 How does the platform know which services are available?	1269
<b>Annex AA (normative): DVB-HTML 1.0 DTD</b>	<b>1270</b>
AA.1 DVB-HTML 1.0 DTD	1270
AA.1.1 DVB-HTML DTD driver	1270
AA.1.2 DVB-HTML DVB Intrinsic Events module	1274
AA.1.3 DVB-HTML Qualified Names module	1275
AA.1.4 DVB-HTML Content Model module	1276

<b>Annex AB (normative):</b>	<b>DVB HTML StyleSheet</b> .....	<b>1280</b>
<b>Annex AC (normative):</b>	<b>ECMAScript Binding</b> .....	<b>1283</b>
AC.1	ECMAScript language binding .....	1283
AC.1.1	Shortcuts to access objects .....	1283
AC.1.2	Grouping the objects of a form .....	1284
AC.2	The DVB-HTML host objects .....	1284
AC.2.1	Object DVBHTMLCollection .....	1284
AC.2.2	Object DVBHTMLDocument .....	1284
AC.2.3	Object DVBHTMLElement .....	1285
AC.2.4	Object DVBHTMLFormElement .....	1286
AC.2.5	Object DVBHTMLSelectElement .....	1286
AC.2.6	Object DVBHTMLOptionElement .....	1287
AC.2.7	Object DVBHTMLInputElement .....	1287
AC.2.8	Object DVBHTMLTextAreaElement .....	1288
AC.2.9	Object DVBHTMLButtonElement .....	1289
AC.2.10	Object DVBHTMLAnchorElement .....	1289
AC.2.11	Object DVBHTMLImageElement .....	1290
AC.2.12	Object DVBHTMLObjectElement .....	1290
AC.2.13	Object DVBHTMLMapElement .....	1291
AC.2.14	Object DVBHTMLAreaElement .....	1291
AC.2.15	Object DVBHTMLFrameSetElement .....	1291
AC.2.16	Object DVBHTMLFrameElement .....	1291
AC.2.17	Object DVBHTMLIFrameElement .....	1292
AC.3	DVB-HTML event host objects .....	1292
AC.3.1	Object DVBLifecycleEvent .....	1292
AC.4	DVB-HTML environment host objects .....	1293
AC.4.1	Navigator Object .....	1293
AC.4.2	Window Object .....	1293
AC.4.3	Location object .....	1294
AC.5	DVB-HTML CSS host objects .....	1294
AC.5.1	DVBInlineStyle .....	1294
AC.5.2	DVBCSSStyle .....	1294
AC.5.3	DVBCSSViewportRule .....	1295
AC.5.4	DVBCSSViewportProperties .....	1295
<b>Annex AD (normative):</b>	<b>Support for DVB-HTML</b> .....	<b>1296</b>
AD.1	Java bindings to DVB extensions .....	1296
AD.1.1	The org.dvb.dom.dvbhtml package .....	1296
AD.1.1.1	DVBHTMLButtonElement .....	1296
AD.1.1.2	DVBHTMLCollection .....	1296
AD.1.1.3	DVBHTMLDocument .....	1296
AD.1.1.4	DVBHTMLElement .....	1297
AD.1.1.5	DVBHTMLFormElement .....	1297
AD.1.1.6	DVBHTMLSelectElement .....	1297
AD.1.1.7	DVBHTMLOptionElement .....	1298
AD.1.1.8	DVBHTMLInputElement .....	1298
AD.1.1.9	DVBHTMLTextAreaElement .....	1299
AD.1.1.10	DVBHTMLAnchorElement .....	1299
AD.1.1.11	DVBHTMLImageElement .....	1300
AD.1.1.12	DVBHTMLObjectElement .....	1300
AD.1.1.13	DVBHTMLMapElement .....	1301
AD.1.1.14	DVBHTMLAreaElement .....	1301
AD.1.1.15	DVBHTMLFrameSetElement .....	1301
AD.1.1.16	DVBHTMLFrameElement .....	1301
AD.1.1.17	DVBHTMLIFrameElement .....	1302
AD.1.2	The org.dvb.dom.css package .....	1302
AD.1.2.1	DVBInlineStyle .....	1302
AD.1.2.2	DVBCSSStyle .....	1303

AD.1.2.3	DVBCSSViewportRule . . . . .	1303
AD.1.2.4	DVBCSSViewportProperties. . . . .	1303
AD.1.3	The org.dvb.dom.environment package . . . . .	1304
AD.1.3.1	Navigator . . . . .	1304
AD.1.3.2	Window . . . . .	1304
AD.1.3.3	Location . . . . .	1304
AD.1.4	The org.dvb.dom.event package . . . . .	1305
AD.1.4.1	DVBLifecycleEvent. . . . .	1305
	DocumentAction . . . . .	1307
	DocumentFactory. . . . .	1308
	MultipleDocumentsAction. . . . .	1310
	HTMLApplication . . . . .	1312
	HTMLInnerApplicationContainer . . . . .	1313
<b>Annex AE (normative): Inner Applications. . . . .</b>		<b>1314</b>
	DVBScene . . . . .	1316
	InnerApplication . . . . .	1320
	InnerApplicationContainer. . . . .	1321
	InnerApplicationEvent. . . . .	1326
	InnerApplicationListener . . . . .	1327
<b>Annex AF (normative): Plug-in APIs. . . . .</b>		<b>1328</b>
	ApplicationSecurityContext. . . . .	1330
	InvalidApplicationException . . . . .	1332
	InvocationHandler . . . . .	1333
	Plugin . . . . .	1334
	Proxy . . . . .	1336
	UndeclaredThrowableException . . . . .	1341
	XletContainer . . . . .	1343
	XletSystemCall . . . . .	1345
<b>Annex AG (normative): Stored application APIs . . . . .</b>		<b>1347</b>
	ApplicationDownloadException . . . . .	1349
	ApplicationStorage. . . . .	1350
	ApplicationStoragePermission. . . . .	1353
	ExtendedAppAttributes . . . . .	1355
	InvalidApplicationException . . . . .	1356
	InvalidDescriptionFileException. . . . .	1357
	NotEnoughResourcesException. . . . .	1358
	ServiceAlreadyExistsException. . . . .	1359
	StoredApplicationService . . . . .	1360
	StoredServiceCreator . . . . .	1363
	UserRejectedInstallException . . . . .	1364
<b>Annex AH (normative): Internet client APIs . . . . .</b>		<b>1365</b>
	BookmarkExistsException . . . . .	1369
	CancelledByUserEvent . . . . .	1370
	ClientNotRunningException . . . . .	1371
	EmailClient . . . . .	1372
	EmailClientService . . . . .	1373
	EntryExistsException. . . . .	1374
	HomePagePermission . . . . .	1375
	InternetClient . . . . .	1376
	InternetClientEvent . . . . .	1378
	InternetClientFailureEvent. . . . .	1379
	InternetClientListener . . . . .	1380
	InternetClientService . . . . .	1381
	InternetClientSuccessEvent . . . . .	1383
	InternetServiceFilter. . . . .	1384
	InternetServiceType . . . . .	1386

PermissionDeniedEvent.....	1387
URLUnavailableEvent.....	1388
UsenetClient.....	1389
UsenetClientService.....	1391
WWWBrowser.....	1392
WWWBrowserService.....	1393
<b>Annex AI (normative): Non-CA Smartcard APIs .....</b>	<b>1396</b>
SmartCardPermission.....	1398
Index.....	1399
History .....	1447

---

# Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: "*Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards*", which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (<http://webapp.etsi.org/IPR/home.asp>).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

---

## Foreword

This Technical Specification (TS) has been produced by the Joint Technical Committee (JTC) Broadcast of the European Broadcasting Union (EBU), Comité Européen de Normalisation ELECTrotechnique (CENELEC) and the European Telecommunications Standards Institute (ETSI).

NOTE: The EBU/ETSI JTC Broadcast was established in 1990 to co-ordinate the drafting of standards in the specific field of broadcasting and related fields. Since 1995 the JTC Broadcast became a tripartite body by including in the Memorandum of Understanding also CENELEC, which is responsible for the standardization of radio and television receivers. The EBU is a professional association of broadcasting organizations whose work includes the co-ordination of its members' activities in the technical, legal, programme-making and programme-exchange domains. The EBU has active members in about 60 countries in the European broadcasting area; its headquarters is in Geneva.

European Broadcasting Union  
CH-1218 GRAND SACONNEX (Geneva)  
Switzerland  
Tel: +41 22 717 21 11  
Fax: +41 22 717 24 81

Founded in September 1993, the DVB Project is a market-led consortium of public and private sector organizations in the television industry. Its aim is to establish the framework for the introduction of MPEG-2 based digital television services. Now comprising over 200 organizations from more than 25 countries around the world, DVB fosters market-led systems, which meet the real needs, and economic circumstances, of the consumer electronics and the broadcast industry.

---

## 0 Introduction

### 0.1 Purpose

The DVB system already provides a comprehensive toolbox to enable interoperable digital video broadcasting systems based on MPEG-2 standards for various transmission media including satellite, cable, terrestrial and microwave. This toolbox also covers interactive services using different kinds of return channels and further supporting functionalities such as service information and many others.

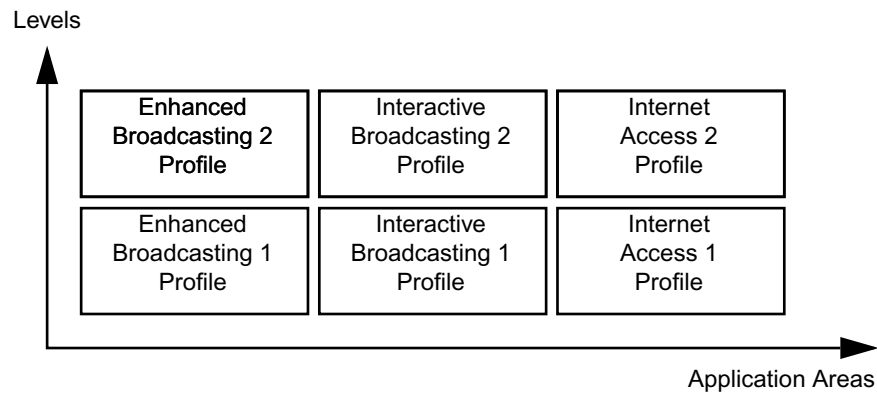
The Multimedia Home Platform (MHP) adds a technical solution for the user terminal that enables the reception and presentation of applications in a vendor, author and broadcaster neutral framework. Here "neutral" includes scenarios that consider legacy infrastructure. Applications from various service providers will be interoperable with different MHP implementations in an horizontal market, where applications, networks, and MHP terminals can be made available by independent providers.

## 0.2 Application Areas

At the beginning the following application areas are considered - Enhanced Broadcasting, Interactive Broadcasting and Internet Access. Enhanced Broadcasting combines digital broadcast of audio/video services with downloaded applications which can enable local interactivity. It does not need an interaction channel. The application area Interactive Broadcasting enables a range of interactive services associated or independent from broadcast services. This application area requires an interaction channel. The application area of Internet Access is intended for the provisioning of Internet services. It also includes links between those Internet services and broadcast services.

## 0.3 Profiles

As not all MHP implementations will be able to support all application areas and as there is a further evolution expected over time, different profiles of the MHP are considered. For the first release of the MHP specification, profiles are mapped to the above mentioned application areas.



**Figure 1 : Application areas and levels of profiles**

Fig. 1 shows six example profiles, derived from two levels for each of the three application areas. The specific definition of the profiles and the particular backward and cross compatibility between profiles is provided in the detailed profile definition chapter of the MHP specification. The following initial definitions apply: <profile><n+1> shall be a strict superset of <profile><n>, and Interactive Broadcasting Profile 1 is defined as a strict superset of Enhanced Broadcasting Profile 1. Other dependencies are left to the detailed definition of future profiles.

## 1 Scope

The present document defines the DVB solution for Multimedia Home Platforms (MHPs) that was developed to fulfil the related DVB commercial requirements [MHP045 \[A\]](#). It relies on the use of appropriate DVB specifications for digital video broadcast and associated interactive services [TR 101 200 \[47\]](#). The MHP is applicable to all DVB defined transmission media and networks such as satellite, cable, terrestrial, microwave.

The final DVB MHP solution is intended to cover the whole range of implementations including Integrated Receiver Decoders (IRDs), integrated TV sets, multimedia computers and local clusters of such devices connected via In-Home Digital Networks (IHDN). This first release focuses on single MHP terminals and does not include such local clusters. Chapters 1-14 specify the applicable technologies and technical definitions in a generic way. Chapter 15 provides detailed profile definitions for the initial profiles Enhanced Broadcasting 1, Interactive Broadcasting 1 and Internet Access 1, which can be extended with future additional profile definitions.

This specification is firstly intended for implementers of MHPs on various hardware and software platforms. Secondly it is intended for developers of applications that use the MHP functionality and APIs.

The MHP specification aims to ensure interoperability between MHP applications and different MHP implementations. Implementers should consult the publisher of this specification regarding conformance.

NOTE: This specification defines the interfaces visible to applications. Application developers should not assume that other related interfaces are available unless they are specifically listed.

## 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.
- A non-specific reference to an ETS shall also be taken to refer to later versions published as an EN with the same number.

Some known errata in these references are identified in [A, "\(normative\): External references; errata, clarifications and exemptions" on page 370](#). These errata take precedence over the published reference.

The following comments apply to particular sources of documents:

[1]	Where the reference is to an ISO specifications it is considered to be a "non-specific" reference additionally officially published amendments and corrigenda are considered to automatically update the referenced document.
[2]	Where an ISBN number is provided for a referenced document it is considered to be "specific reference".
[3]	References to RFCs are considered to be "specific references". An RFC being indicated obsoleted by another RFC is not considered significant.
[4]	URL references with note <a href="#">[4]</a> are provided for convenience to access the document in electronic form.
[5]	URL references with note <a href="#">[5]</a> are the normative method to access the reference



[6]	ETSI specifications are available from the ETSI server at: <a href="http://www.etsi.org">http://www.etsi.org</a> . However, the ETSI server provides the current edition of the specification and in every case this specification makes "specific" references which in the future may not be the current reference.
[7]	The Sun Specifications for DVB are available from: <a href="http://java.sun.com/products/specformhp/">http://java.sun.com/products/specformhp/</a> or on CD-ROM (ISBN 1-892488-25-6) published by: Sun Microsystems MS USCA14-103 Palo Alto CA 94303 USA e-mail: docs@java.sun.com Phone: +1 408 276-7426

	Reference	Edition	Description	Note
[1]	CIE 15	2nd Edition, 1986 ISBN 3 900 734 00 3	Colorimetry, CIE, Vienna	[2]
[2]	CORBA/IIOP	2.1	The Common Object Request Broker: Architecture and Specification, Object Management Group. <a href="ftp://ftp.omg.org/pub/docs/formal/97-09-01.pdf">ftp://ftp.omg.org/pub/docs/formal/97-09-01.pdf</a>	[5]
[3]	DAVIC 1.4.1p9	June 1999	DAVIC 1.4.1 Specification Part 9, Complete DAVIC Specifications, DAVIC. <a href="http://www.davic.org">http://www.davic.org</a>	[5]
[4]	EN 300 468	1.3.1	Digital broadcasting systems for television, sound and data services; Specification for Service Information (SI) in Digital Video Broadcasting (DVB) systems	[6]
[5]	EN 301 192	1.2.1	Specification for Data Broadcast	[6]
[6]	EN 301 193	1.1.1	DVB Interaction Channel through DECT	[6]
[7]	EN 301 195	1.1.1	DVB Interaction Channel through GSM	[6]
[8]	EN 301 199	1.2.1	DVB Interaction channel for LMDS distribution systems	[6]
[9]	TR 101 154	1.4.1	DVB Implementation Guidelines for the use of MPEG-2 Systems, Video and Audio in Satellite, Cable and Terrestrial Broadcasting Applications.	[6]
[10]	ETR 162		Digital broadcasting systems for television, sound and data services; Allocation of Service Information (SI) codes for Digital Video Broadcasting (DVB) systems	[6]
[11]	ETR 211	2	Digital Video Broadcasting (DVB); Guidelines on implementation and usage of Service Information (SI)	[6]
[12]	EN 300 472	1.2.2	DVB Specification for conveying ITU-R System B Teletext in DVB bitstreams.	[6]
[13]	EN 300 743	1	Digital Video Broadcasting (DVB), DVB subtitling	[6]
[14]	ETS 300 800	1	DVB Interaction Channel for Cable TV distribution systems	[6]
[15]	ETS 300 801	1	DVB Interaction Channel through PSTN/ISDN	[6]
[16]	ETS 300 802	1	Network Independent Protocols for Interactive Services	[6]
[17]	GIF 89a		GRAPHICS INTERCHANGE FORMAT(sm) Version 89a, (c)1987, 1988, 1989, 1990 Copyright CompuServe Incorporated Columbus, Ohio <a href="http://www.w3.org/Graphics/GIF/spec-gif89a.txt">http://www.w3.org/Graphics/GIF/spec-gif89a.txt</a>	

Reference	Edition	Description	Note	
[18]	ISO 10646-1	Information technology - Universal multiple-octet coded character set (UCS), part 1: Architecture and Basic Multilingual Plane" (also known as Unicode, UTF)	[1]	
[19]	ISO 639.2	Code for the representation of names of languages	[1]	
[20]	ISO 8859	Information processing - 8-bit single-byte coded graphic character sets, Latin alphabets	[1]	
[21]	ISO/IEC 10918-1	Digital compression encoding of continuous-tone still images (JPEG). <a href="http://www.w3.org/Graphics/JPEG/itu-t81.pdf">http://www.w3.org/Graphics/JPEG/itu-t81.pdf</a>	[1] [4]	
[22]	ISO/IEC 11172-3	1993	Information technology—Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s—Part 3: Audio (Note: known as MPEG-1 Audio)	[1]
[23]	ISO/IEC 13818-1	1996	Information technology - Generic coding of moving pictures and associated audio information: Systems.	[1]
[24]	ISO/IEC 13818-2	1996	Information technology—Generic coding of moving pictures and associated audio information—Part 2: Video (MPEG-2 Video)	[1]
[25]	ISO/IEC 13818-3	2 <sup>nd</sup> Ed. 1998	Information technology - Generic coding of moving pictures and associated audio - Part 3, MPEG-2 Audio.	[1]
[26]	ISO/IEC 13818-6	1998	Information technology -Generic coding of moving pictures and associated audio information: Extensions for Digital Storage Media Command and Control.	[1]
[27]	IEC 61966-2-1	1	Multimedia systems and equipment - Colour measurement and management - Part 2-1: Colour management - Default RGB colour space - sRGB	
[28]	ITU-R BT.470	6	Conventional television systems	
[29]	ITU-R BT.601	5	Studio Encoding Parameters of Digital Television for standard 4:3 and Wide-Screen 16:9 aspect ratios	
[30]	ITU-R BT.709	3	Parameter values for the HDTV standards for production and international programme exchange	
[31]	JAE 1.1.8 API	Part of ISBN:1-892488-25-6	Java Platform 1.1 API Specification	[7]
[32]	Java Language Spec	ISBN 0-201-63451-1	The Java Language Specification by James Gosling, Bill Joy and Guy Steele. <a href="ftp://ftp.javasoft.com/docs/specs/langspec-1.0.pdf">ftp://ftp.javasoft.com/docs/specs/langspec-1.0.pdf</a> including the clarifications at: <a href="http://java.sun.com/docs/books/jls/clarify.html">http://java.sun.com/docs/books/jls/clarify.html</a>	[2] [4]
[33]	Java Media Player Specification	Part of ISBN:1-892488-25-6	Java Media Framework API Version 1.0 specification	[7]
[34]	Java VM	ISBN: 0-201-63452-X	The Java Virtual Machine Specification, T. Lindholm and F. Yellin, Addison-Wesley.	[2]
[35]	JFIF		JPEG File Interchange Format, Eric Hamilton, C-Cube Microsystems. <a href="http://www.w3.org/Graphics/JPEG/jfif3.pdf">http://www.w3.org/Graphics/JPEG/jfif3.pdf</a>	[5]
[36]	PersonalJAE	Part of ISBN:1-892488-25-6	The OEM PersonalJava Application Environment Version 1.2a specification	[7]
[37]	PNG	V1 01-Oct-96	Portable Network Graphics. Available at <a href="http://www.w3.org/TR/REC-png.html">http://www.w3.org/TR/REC-png.html</a>	[5]

	Reference	Edition	Description	Note
[38]	RFC 1321	April 1992	The MD5 Message-Digest Algorithm	[3]
[39]	RFC 1990	August 1996	(MP) "The PPP Multilink Protocol", K. Sklower, B. Lloyd, G. McGregor, D. Carr, T. Coradetti.	[3]
[40]	RFC 2616	June 1999	IETF Hypertext Transfer Protocol -- HTTP/1.1	[3]
[41]	RFC 2396	August 1998	IETF Uniform Resource Identifiers (URI): Generic Syntax	[3]
[42]	RFC 768	28.08.1980	(UDP) "User Datagram Protocol", J. Postel.	[3]
[43]	RFC 791	01.09.1981	(IP) "Internet Protocol", J. Postel.	[3]
[44]	RFC 793	01.09.1981	(TCP) "Transmission Control Protocol", J. Postel.	[3]
[45]	RFC 1112	August 1989	IETF Host extensions for IP multicasting.	[3]
[46]	TR 101 194	1.1.1	Guidelines for the use of ETS 300 802	
[47]	TR 101 200	1.1.1	"Digital Video Broadcasting (DVB); Guideline for the use of DVB specifications and standards".	
[48]	TR 101 201	1.1.1	DVB Interaction channel for SMATV systems	
[49]	TR 101 202	1.1.1	Guidelines for the use of EN 301 192	
[50]	HAVi		See annex V, "(normative): HAVi Level 2 User Interface" on page 790.	
[51]	Java TV	Part of ISBN:1-892488-25-6	Java TV API Version 1.0 specification	[7]
[52]	Hunt, R.W.G.	1987	Measuring Colour, Ellis Horwood Series in Applied Science and Industrial Technology. Ellis Harwood Limited, Chichester, England.	
[53]	ITU-T X.501	1993	ITU-T Recommendation X.501: Information Technology - Open Systems Interconnection - The Directory: Models, 1993.	
[54]	ITU-T X.509	08/97	Information technology - Open Systems Interconnection - The Directory: Authentication framework	
[55]	ITU-T X.520	11/93	Information Technology Open Systems Interconnection The Directory: Selected Attribute Types	
[56]	RFC 2313	March 98	PKCS #1: RSA Encryption Version 1.5	[3]
[57]	ASN.1	07/94	ITU-T X.680 "Information Technology Abstract Syntax Notation One (ASN.1): Specification Of Basic Notation"  and  ITU-T X.690 "Information Technology ASN.1 Encoding Rules: Specification Of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) And Distinguished Encoding Rules (DER)"	
[58]	RFC 2459	January 1999	Internet X.509 Public Key Infrastructure. Certificate and CRL Profile.	[3]
[59]	POSIX	ISBN:1-55937-255-9	IEEE Standard for Information Technology - Portable Operating System Interface (POSIX) - Part 2: Shell and Utilities (Vol. 1);  IEEE Std 1003.2-1992; The Institute of Electrical and Electronics Engineers; New York; 1993.	
[60]	JSSE	Part of ISBN:1-892488-25-6	Java Secure Sockets Extension API Specification Version 1.0.2	[7]
[61]	ETS 300 706	Edition 1 - 1997-05	Enhanced Teletext Specification	[6]

Reference	Edition	Description	Note
[62]	FIPS-180-1 April 1995	NIST, FIPS PUB 180-1: Secure Hash Standard <a href="http://csrc.nist.gov/fips/fip180-1.txt">http://csrc.nist.gov/fips/fip180-1.txt</a>	[4]
[63]	RFC 2246 January 1999	The TLS Protocol, Version 1.0	[3]
[64]	RFC 2045 November 1996	Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies	[3]
[65]	XML 1.0 Second Edition	Extensible Markup Language (XML) 1.0 <a href="http://www.w3.org/TR/2000/REC-xml-20001006">http://www.w3.org/TR/2000/REC-xml-20001006</a>	
[66]	ISO 639.2	Codes for the representation of names of languages – Part 2: Alpha-3 code.	
[67]	RFC 1738 December 1994	Uniform Resource Locators (URL)	
[68]	JVM Errata Part of ISBN:1-892488-25-6	Errata for the Java Virtual Machine Specification	[7]
[69]	Inner Classes Part of ISBN:1-892488-25-6	Inner Classes Specification	[7]
[70]	EN 50221	Common Interface Specification for Conditional Access and other Digital Video Broadcasting Decoder Applications	
[71]	TS 101 699 1.1.1	Digital Video Broadcasting (DVB); Extensions to the Common Interface Specification	
[72]	JAE 1.1.8 const Part of ISBN:1-892488-25-6	JAE 1.1.8 API Constants	[7]
[73]	JAE 1.2.2 const Part of ISBN:1-892488-25-6	JAE 1.2.2 API Constants	[7]
[74]	JMF const Part of ISBN:1-892488-25-6	Java Media Framework API Version 1.0 Constants	[7]
[75]	RFC 1950 May 1996	ZLIB Compressed Data Format Specification version 3.3	
[76]	RFC 1951 May 1996	DEFLATE Compressed Data Format Specification version 1.3	
[77]	ZIP	PKWare description of the ZIP file format, no title, filename Appnote.txt. <a href="ftp://ftp.pkware.com/appnote.zip">ftp://ftp.pkware.com/appnote.zip</a>	
[78]	RFC 1945 May 1996	Hypertext Transfer Protocol -- HTTP/1.0	
[79]	XHTML™ 1.0: The Extensible HyperText Markup Language	<a href="http://www.w3.org/TR/xhtml1">http://www.w3.org/TR/xhtml1</a>	
[80]	Modularization of XHTML	<a href="http://www.w3.org/TR/xhtml-modularization">http://www.w3.org/TR/xhtml-modularization</a>	
[81]	ECMAScript	'Information technology - ECMAScript language specification' ISO/IEC 16262, First Edition 1998-12-15: ICS 35.060  ECMA (European Computer Manufacturers Association) ECMAScript Language Specification. <a href="http://www.ecma.ch/ecma1/STAND/ECMA-262.HTM">http://www.ecma.ch/ecma1/STAND/ECMA-262.HTM</a>	
[82]	Document Object Model (DOM) Level 2 Core Specification	Document Object Model DOM Level 2 Core: <a href="http://www.w3.org/TR/DOM-Level-2-Core">http://www.w3.org/TR/DOM-Level-2-Core</a>	

Reference	Edition	Description	Note
[83]	Document Object Model (DOM) Level 1 Specification	Document Object Model DOM Level 1: <a href="http://www.w3.org/TR/REC-DOM-Level-1">http://www.w3.org/TR/REC-DOM-Level-1</a>	
[84]	Document Object Model (DOM) Level 2 Events Specification	Document Object Model DOM Level 2 Events: <a href="http://www.w3.org/TR/DOM-Level-2-Events">http://www.w3.org/TR/DOM-Level-2-Events</a>	
[85]	Document Object Model (DOM) Level 2 Style Specification	Document Object Model DOM Level 2 Style: <a href="http://www.w3.org/TR/DOM-Level-2-Style">http://www.w3.org/TR/DOM-Level-2-Style</a>	
[86]	Document Object Model (DOM) Level 2 Views Specification	Document Object Model DOM Level 2 Views: <a href="http://www.w3.org/TR/DOM-Level-2-Views">http://www.w3.org/TR/DOM-Level-2-Views</a>	
[87]	CSS 2	Cascading Style Sheets CSS 2: <a href="http://www.w3.org/TR/REC-CSS2/">http://www.w3.org/TR/REC-CSS2/</a> Including the errata published in REC-CSS2-19980512.	
[88]	RFC2318	The text/css Media Type	
[89]	Namespaces in XML	Namespaces in XML: <a href="http://www.w3.org/TR/REC-xml-names">http://www.w3.org/TR/REC-xml-names</a>	
[90]	HTML 4	4.01	HTML 4.01 Specification W3C Recommendation 24 December 1999 <a href="http://www.w3.org/TR/html401">http://www.w3.org/TR/html401</a>
[91]	SVG	1.0	Scalable Vector Graphics (SVG) 1.0 Specification. <a href="http://www.w3.org/TR/2000/CR-SVG-20000802/">http://www.w3.org/TR/2000/CR-SVG-20000802/</a>
[92]	Embedded OCF	1.2.2	OpenCard Framework for Embedded Devices Specification <a href="http://www.opencard.org/download/docs/1.2T/ocfembeddedspec.zip">http://www.opencard.org/download/docs/1.2T/ocfembeddedspec.zip</a>
[93]	SMTP		RFC 821, Simple Mail Transport Protocol
[94]	NNTP		RFC 977, Network News Transport Protocol
[95]	Namespaces in XML	19990114	REC-xml-names-19990114, Namespaces in XML, World Wide Web Consortium 14-January-1999 <a href="http://www.w3.org/TR/1999/REC-xml-names-19990114">http://www.w3.org/TR/1999/REC-xml-names-19990114</a>
[96]	RFC 2109	February 1997	HTTP State Management Mechanism
[97]	RFC 1123	October 1989	Requirements for Internet Hosts •• Application and Support
[98]	RFC 2818		HTTP over TLS
[99]	Associating Style Sheets with XML documents	1.0	<a href="http://www.w3.org/TR/xml-stylesheet">http://www.w3.org/TR/xml-stylesheet</a>
[100]	RFC 2368	July 1998	The Use of URLs as Meta-Syntax for Core Mail List Commands and their Transport through Message Header Fields

---

## 3 Definitions and abbreviations

### 3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

**API:** Application Program Interface. An interface between an application and a particular feature, function or resource of the MHP.

**application:** A functional implementation realised as software running in one or spread over several interplaying hardware entities.

**application boundary:** A concise general description of the data elements (HTML documents, code files, images etc.) used to form one application and the logical locator of the entry point, the application boundary is described by a regular expression over the URL language. Where no such boundary is drawn, the default boundary shall be the entire set of documents that the MHP platform can access.

**application manager:** The Application Manager is the entity in the MHP that is responsible for managing the lifecycle of the applications in the MHP. It manages both the DVB-J applications and non-DVB-J applications.

**autostart applications:** This terms has different definitions depending on the application format:

A DVB-J autostart application is an application that is automatically loaded and executed by the Application Manager as soon as the user selects a service on which the application is signalled as autostart.

Auto start application a DVB-HTML application in a broadcast stream can be signalled as auto start in the same way that other DVB applications can, but note that it may not actually start providing service until it receives a start trigger.

**character:** A specific "letter" or other identifiable symbol, e.g. "A".

**character encoding:** A character encoding is a mapping between an integer input value, and the textual character that is represented by this mapping, e.g. in ASCII value 65 (decimal) is character "A", or shift-JIS for Japanese characters.

**character set:** See character encoding.

**communications network:** A system of interconnected entities providing data interchange between points or from a point to multiple points.

**definite:** A resource is considered of indefinite duration if the exact end point is not known in advance of accessing the resource (e.g. Transport Streams in MPEG-2). A resource is considered of definite duration if in principle the exact length is known (e.g. a file resource).

**delegated application:** An application encoded in a representation that is not directly consumable by an MHP terminal's standard mechanisms.

**domain of an application:** The domain of an Xlet characterizes the "space" within which the Xlet is able to execute. This includes both the "connection" where the Xlet is delivered and other "connections" where an already executing Xlet is allowed to continue executing.

An application cannot run outside its domain. The maximum lifetime of an application extends from the moment the user navigates to its domain until the moment that the user navigates away from its domain.

In the broadcast case a "connection" corresponds to a DVB-service. Broadcast signalling indicates which services can load an application and which services allow an already active application to continue.

**DVB network:** A collection of MPEG-2 Transport Stream multiplexes transmitted on a single delivery system, e.g. all digital channels on a specific cable system.

**DVB-HTML actor:** A DVB-HTML actor is defined as the locus of activity or process involved in running the specific set of DVB-HTML documents for some DVB-HTML application, plus any instantiated context for that data. The actor runs inside a user agent (native, plug-in or downloaded). The nature of the process is not defined explicitly as it depends on the nature of the user agent itself. More than one such locus of activity may be present in any given user agent.

**DVB-HTML application:** A DVB-HTML application is defined as a set of documents selected from the DVB-HTML family of elements and content formats as defined in the specification. The extent of the set is described by the application boundary.

**DVB-HTML application states:** DVB-HTML application states are logical states that a DVB-HTML actor can be in, (as opposed to states the user agent may be in), these states may have instance data logically associated with them (e.g. the application id and entry point).

**DVB-HTML document:** A complete unit of one the HTML family of elements or content formats defined in this specification.

**DVB-J:** The Java platform defined as part of the MHP specification.

**DVB-J API:** One of the Java APIs standardised as part of the MHP specification.

**DVB-J application:** A DVB-J Application is a set of DVB-J classes that operate together and need to be signalled as a single instance to the Application Manager so that it is aware of its existence and can control its lifetime through a lifecycle interface.

**events:** Asynchronous communication between applications and the MHP on which they are being executed. They provide communication between solution elements.

**font:** A font is a mechanism that allows the specific rendering of a particular character to be specified – e.g. Tiresias, 12 point. In practice a font file format will incorporate some aspects of a character encoding.

**function:** A function is a process which conveys or transforms data in a predictable way. It may be effected by hardware, software or a combination of the two.

**hardware entity:** Is an independent piece of hardware which forms part of a (multiple) local cluster of elements which as a whole is called MHP. A hardware entity is for example: a Set top box, a digital VCR or a conditional access module. A hardware entity includes a number of resources. Each resource provides a number of functions.

**indefinite:** See definite.

**internet access application:** An application which is resident on an MHP terminal and used for presenting content from the internet. Applications which are downloaded to an MHP terminal when required are also covered by this definition only if this downloading process is transparent to any MHP applications. Internet access applications do not include the applications or content obtained from the internet. They are very similar conceptually to a DVB-HTML support application.

**inner application:** Application content that forms part of a larger application and possibly of a different type. For example, an Xlet embedded within an DVB-HTML application.

**interoperable plug-in:** A plug-in that only requires standard MHP APIs.

**interoperability:** The reception and presentation of applications in a vendor, author and broadcaster neutral framework (From MHP45r12).

**java API:** Is a standard interface for use by platform independent application software. It is expressed in the Java language.

**lifetime of an application:** The lifetime of an application characterizes the time from which the application is Loaded to the time the application is Destroyed.

**locator:** This term has different definitions depending on the application format:

A DVB-HTML locator is a link, expressed in the syntax in [RFC 2396 \[41\]](#), which provides an unambiguous pointer to a DVB-HTML document accessible to the MHP in a specific transport stream. The scheme specified should resolve to one of the available transports signalled for the DVB-HTML application. For signed DVB-HTML applications the schemes http and https (ftp, others?) may use the return channel. This version of the specification does not include a scheme for transport independent locators, future versions are expected to do so.

This term in the DVB-HTML context should not be confused with the DVB-J class of the same name.

**MHP:** The Multimedia Home Platform (MHP) consists of an MHP viewer terminal, including all possible low to high functionality implementations, its associated peripherals and the in-home digital network.

**MHP connected resource:** A resource used as part of the MHP which, on its own, is not conformant to the specification but which is connected to an MHP Terminal in such a way that the whole is part of the MHP.

**MHP service:** A logical service in an MHP which can be selected through the service selection API or functional equivalents. This includes broadcast DVB services, stored services and MHP applications executed in response to an AIT file loaded over the interaction channel.

**MHP solution:** The MHP solution encompasses the whole set of technologies necessary to implement the MHP including protocols and APIs.

**MHP terminal:** A single piece of physical equipment conforming to the MHP specification, in particular in that it contains a Virtual Machine and an instance of the MHP API.

**native signalling:** Signalling infrastructure for a delegated application that are not part of an MHP terminal's standard mechanisms.

**navigator:** A resident application, typically provided by the manufacturer, which the end-user can activate at any time. The navigator can be used to select services, applications, and initiate Interoperable applications.

**object carousel:** A repetitively broadcast file system.

**OID:** X.509 Object Identifier.

**persistent storage:** Memory available in the MHP which can be read/written to by an application and which may outlive the application's own life.

Persistent storage may be volatile or non-volatile.

**plug-in:** A set of functionality which can be added to a generic platform in order to provide interpretation of DVB registered, but non-DVB-J, application formats; e.g. HTML3.2 or MHEG-5.

**plug-in application:** An application that conforms to an application format for which a plug-in has been registered with DVB and which is only interoperable within terminals which have the appropriate plug-in resident or connected to networks where an appropriate Plug-In is being broadcast.

**profile:** A description of a series of minimum configurations, defined as part of the specification, providing different capabilities of the MHP. It maps a set of functions which characterise the scope of service options. The number of profiles is small. The mapping of functions into resources and subsequently into hardware entities is out of the scope of the specification and is left to manufacturers.

**regular expression:** A method of capturing a large, possibly infinite set of strings in a compact representation.

**resident application:** An application available from non-volatile storage in an MHP device which may be expressed in DVB-J but need not be so. Its delivery route is not specified.

**resource:** Is a well defined capability or asset of a system entity, which can be used to contribute to the realisation of a service. Examples: MPEG decoder, Graphics system.

**return channel:** The communications mechanism which provides connection between the MHP and a remote server.

**runtime code extension:** The ability to extend, at runtime, the executable code of an application with code not originally packaged with the application. This facility is provided, for example, by `org.dvb.lang.DVBClassLoader` in DVB-J and by `eval()` in ECMAScript.

**sandbox:** Unsigned applications and signed applications without a permission file have access to all the APIs for which there is no permission signalling defined. This is commonly called the sandbox.

**service:** A sequence of programs under the control of a broadcaster which can be broadcast as part of a schedule.



**service component:** A part of a service. For a DVB service, service components are normally the MPEG elementary streams listed in the PMT for the service. Where multiple streams of subtitles or MPEG-2 audio representing different languages are carried in the same MPEG elementary stream these are logically service components but are not exposed through the MHP APIs as being distinct and separate.

**stand-alone application:** An application which is not related to a conventional DVB service. In this version of this specification, this includes applications running as part of a stored service and applications whose signalling was carried over the interaction channel.

**stream:** A unidirectional continuous flow of content. Example: MPEG2 video.

**stored service:** A set of MHP applications stored within the MHP terminal and treated as a service with regard to life cycle, service selection etc.

**system software:** Software implementation below the API for a specific platform entirely under control of the manufacturer.

**trigger:** A trigger is an event that may cause a change in the behaviour of a DVB-HTML application that registers interest in such events. Triggers may come from many sources e.g. the broadcast stream, or may be generated from other data (such as the system clock), or may be generated as a result of user interaction. The trigger may include a reference to time, which may be absolute (UTC), relative to some other event, relative to the NPT of a media stream. It also can carry some semantically significant payload in order to affect changes in an application based on information not available at the time an application was written.

**tuning:** the act of switching between two MPEG transport streams or multiplexes. Switching between two DVB services carried in the same transport stream is not tuning.

**viewer:** End-user of the MHP terminal.

**xlet:** Interface used for DVB-J application life cycle control.

## 3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

API	Application Programming Interface
AV	Audio Video
AWT	Abstract Windowing Toolkit
CA	Conditional Access
CI	Common Interface
CLUT	Colour Lookup Table
CSS	Cascading Style Sheets
DAVIC	Digital Audio Visual Council
DCT	Discrete Cosine Transformation
DECT	Digital Enhanced Cordless Telecommunications
DOM	Document Object Model
DSM-CC	Digital Storage Media - Command and Control
DSM-CC-OC	Digital Storage Media - Command and Control Object Carousel
DSM-CC-UU	Digital Storage Media - Command and Control User to User
DVB	Digital Video Broadcasting
ECMA	European Computer Manufacturers Association
EPG	Electronic Program Guide
ETSI	European Telecommunications Standards Institute
GIF	Graphics Interchange Format
GSM	Global System for Mobile communications

GUI	Graphical User Interface
HTML	Hyper Text Mark-up Language
HTTP	Hyper Text Transport Protocol
I/O	Input / Output
IHDN	In Home Digital Network
IP	Internet Protocol
IPR	Intellectual Property Rights
IRD	Integrated Receiver Decoder
ISDN	Integrated Services Digital Network
ISO	International Standardisation Organisation
ITU	International Telecommunication Union
JDK	Java Development Kit
JFIF	JPEG File Interchange Format
JMF	Java Media Framework
JPEG	Joint Picture Expert Group
LMDS	Local Multipoint Distribution System
MHEG	Multimedia Hypermedia Expert Group
MHP	Multimedia Home Platform
MMDS	Multipoint Microwave Distribution System
MPEG	Moving Picture Expert Group
OC	Object Carousel
OS	Operating System
OSD	On Screen Display
PFR	Portable Font Resource
PMT	Program Map Table
PNG	Portable Network Graphics
PSI	Program Specific Information
PSTN	Public Switched Telephone Network
RAM	Random Access Memory
ROM	Read Only Memory
SI	Service Information
SMATV	Satellite Master-Antenna Television
TCP	Transmission Control Protocol
TS	Transport Stream
UCS	Universal Multiple-Octet Coded Character Set
UDP	User Datagram Protocol
UI	User Interface
URL	Uniform Resource Locator
UTF	UCS Transformation Coding
UU	User to User
VM	Virtual Machine
WAN	Wide Area Network

---

## 4 Conventions

Void

## 5 Basic Architecture

### 5.1 Context

At its simplest level, the MHP is set in the following context (see figure 2). The software of the MHP has access to flows of streams and data, and may write some data to storage. The platform may be able to route streams and data outwards to a sink or store.

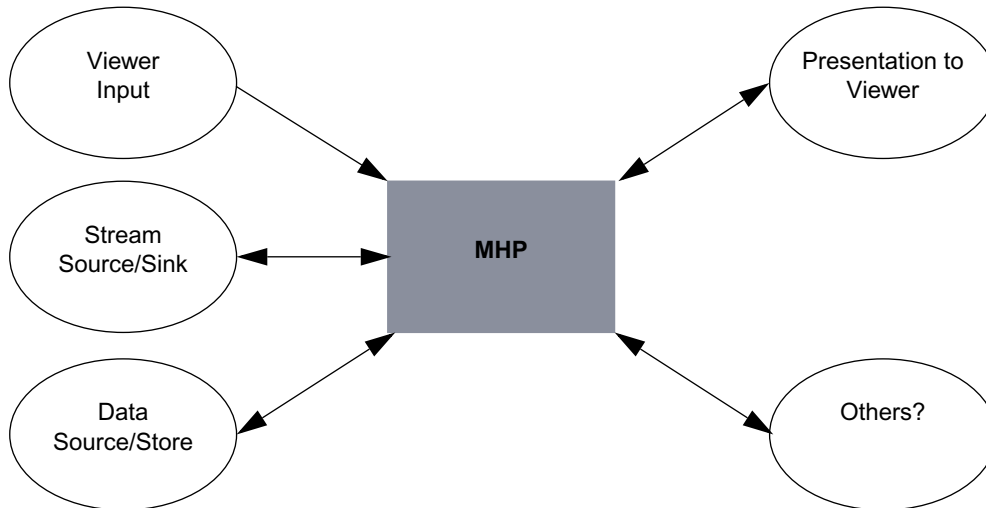


Figure 2 : MHP context

The platform will receive inputs from Viewer input devices and output communications through a screen or other outputs like loudspeakers to present to the viewer. The platform may have access to communications with remote entities.

The diagram in figure 3 shows a possible set of external interfaces between an MHP and the outside world. This is one example only but it serves to illustrate a series of principles.

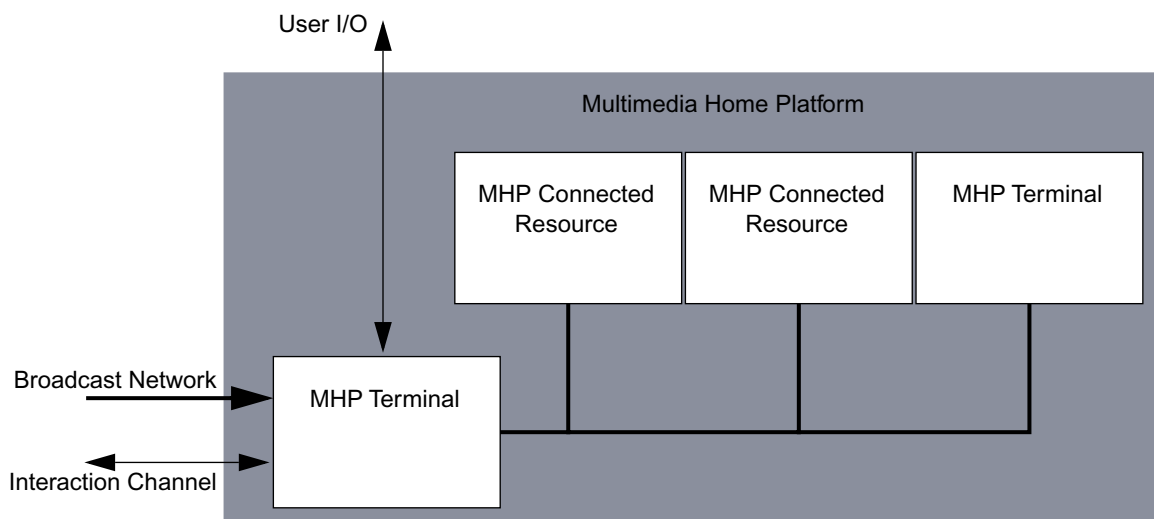


Figure 3 : External interfaces between an MHP and the outside world

The resources of the MHP, accessible by an application, may be contained in a series of different but connected physical entities.

The local cluster may connect a number of MHP terminals and resources

A cluster may also include resources which are not part of the MHP infrastructure and are not available to the application.

The local cluster is understood to be consistent with the DVB IHDN specification. The detailed description of the MHP in the local cluster is not in the first version of the specification.

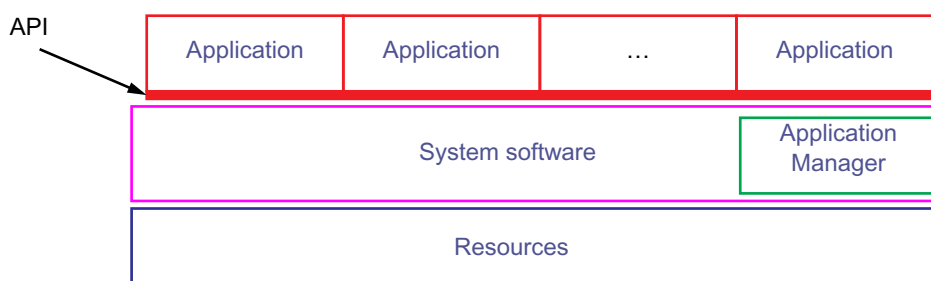
## 5.2 Architecture

The Architecture describes how the MHP software elements are organized.

The MHP model considers 3 layers (figure 4):

- Resources
- System software
- Applications

The API lies between the Applications and the System Software seen from the perspective of an application.



**Figure 4 : Basic architecture**

### 5.2.1 Resources

The hardware entities in the platform include a number of functions. They are represented by hardware or software resources. There is no assumption about how they are grouped. The model considers that there can be more than one hardware entity in the total Platform.

From an abstract point of view it makes no difference if the logical resources are mapped into one or several hardware entities. What is important is that resources are provided to the MHP transparently. An application should be able to access all locally connected resources as if they were elements of a single entity.

### 5.2.2 System software

Applications will not directly address resources. The system software brings an abstract view of such resources. This middle layer isolates the application from the hardware, enabling portability of the application.

The implementations of the Resources and System software are not specified in this document.

#### 5.2.2.1 Application Manager

The system software includes an application management function, which is responsible for managing the lifecycle of all applications, including Interoperable ones.

### 5.2.3 Application

Applications implement interactive services as software running in one or more hardware entities. The interface for MHP applications is a top view from application to the system software.

Figure 4 on page 49 illustrates an idealised architecture model of the processes which will occur in an MHP. A hierarchy of control is assumed in which each layer controls the processes in adjacent layers. The top layer is responsible for the control of the operation via interactive applications. The Application Manager is part of the System Software and as such is implementation specific. It interacts with all applications.

The System Software implements the API by presenting an abstract model of:

- Streams played from different sources and pipes for conducting them.
- Commands and events
- Data records or files
- The hardware resources

The API provides the associated services to applications.

In fact there are many APIs which implement distinct services and interfaces. These are described in detail in the specification, either by reference to external documents, or by detailed specification.

The specification describes the interfaces between the network, the application and the system software of the MHP terminal. The implementation of any of these three is not specified in this document.

## 5.3 Interfaces Between an MHP Application and the MHP System

Application(s) use the API to access the actual resources of the receiver, including: databases, streamed media decoders, static content decoders and communications. These resources are functional entities of the receiver and may be finally mapped onto the hardware of the receiver in some manner.

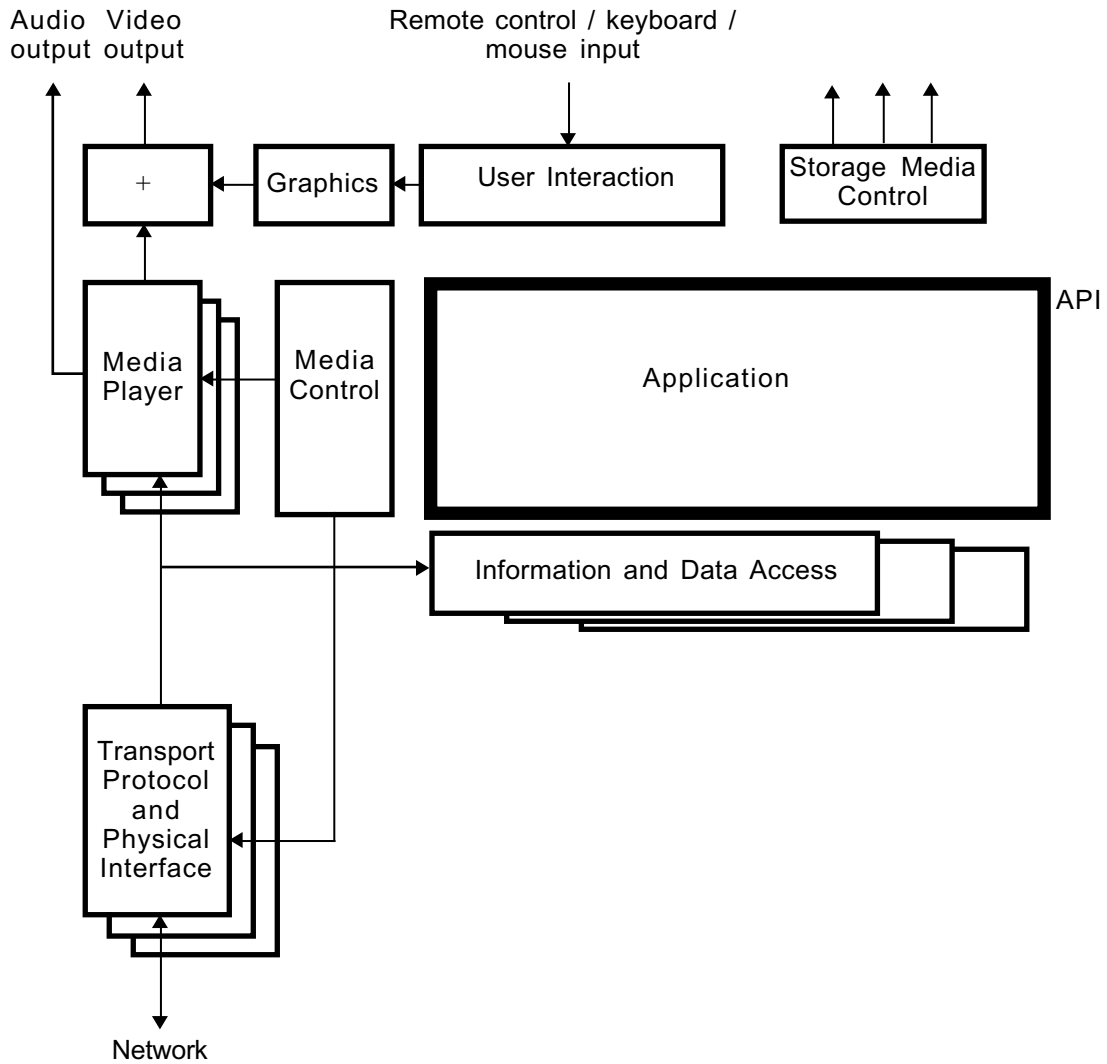
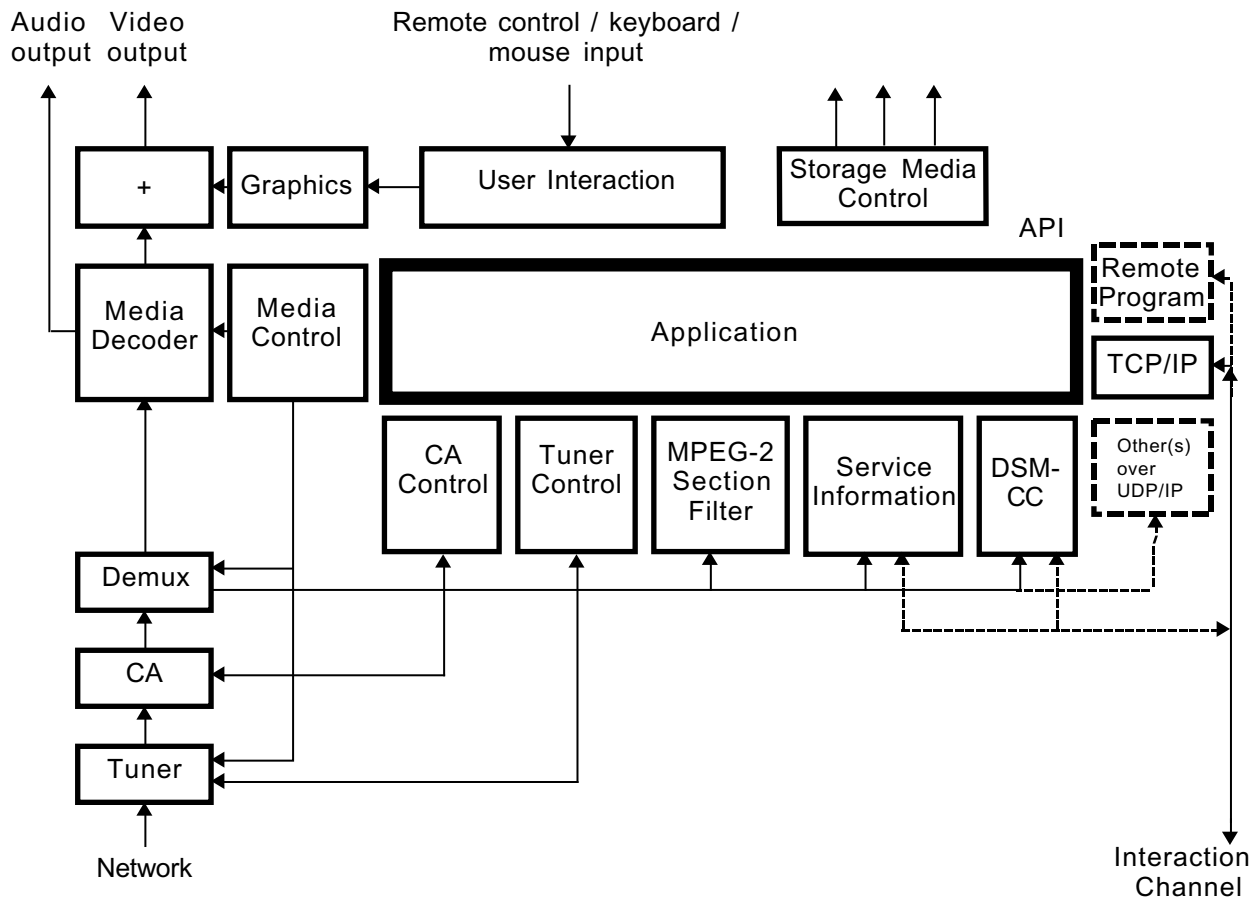


Figure 5 : Interfaces between an MHP application and the MHP system



**Figure 6 : Interfaces between an MHP application and the MHP system with more details**

The diagrams in figure 5 and figure 6 show these interfaces and their relationships to media and information flows within an MHP system - excluding any local cluster issues. The first diagram shows a generic MHP, the second a specific instance of an enhanced broadcast or interactive TV profile system, with optional additions shown.

In figure 5 and figure 6, only the border between the application and the rest of the system is in the scope of this specification. All the rest of each diagram is implementation dependent and shown for information purposes only.

## 5.4 Plug-ins

A "plug-in" is a set of functionality that can be added to a generic platform in order to provide interpretation of application and content formats not specified by this specification to be included in MHP terminals.

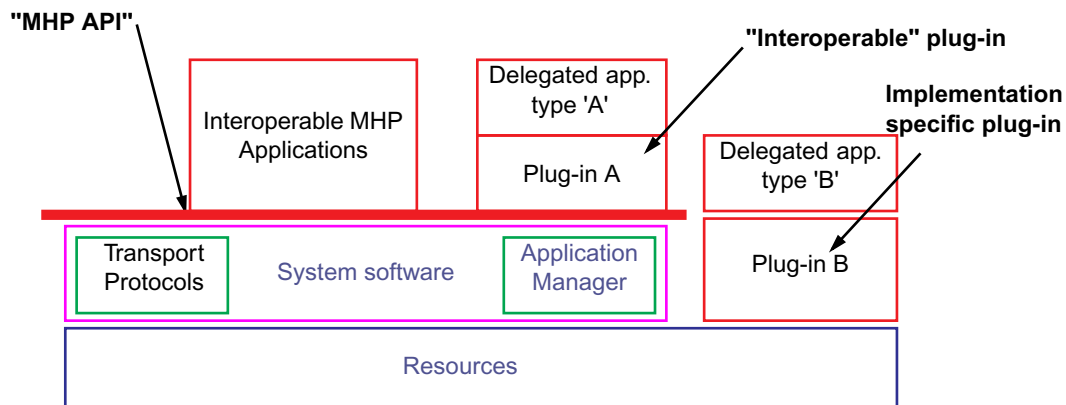
**NOTE:** Those organisations concerned with interoperation between the standard MHP platform and other platforms need to specify the plug-in properly for such platforms.

The choice of which plug-ins to use must be in the hands of the end-user in order that he can have a choice of sources of service. This option can be exercised in a number of ways, including the purchase of equipment with "built-in" plug-in functionality, the positive selection of a download, or the automatic selection of a download where there is no memory resource limitation.

The plug-in application may stay resident where the design of the platform implementation allows. The MHP including the plug-in must behave, once the plug-in is loaded and operational, in the same way as a platform supporting the format of the delegated applications without the use of a plug-in.



Figure 7 illustrates the position of two types of Plug-in in the MHP.



**Figure 7 : Illustrative plug-in implementation options**

There are two possible types of Plug-in implementation:

- Using implementation-specific code (e.g. in native code, or using implementation-specific Java APIs). This is called an implementation-specific plug-in, illustrated as plug-in "B" in the figure.
- An MHP application. This is called an interoperable plug-in, illustrated as plug-in "A" in the figure.

The internal specification of both plug-ins ("A" and "B") is outside of the scope of this specification. They are illustrated to show their relationship to the platform.

### 5.4.1 Security Model

Plug-ins must have sufficient access to the resources in the platform to implement the specification concerned. An implementation-specific plug-in may have access to many of the resources of the platform irrespective of the MHP security model. All plug-ins are responsible for managing the security of the applications they execute.

If a plug-ins needs privileged access to resources not available to all downloaded applications (i.e. not in the "sand box") in order to provide equivalent function to the "legacy" supported they will require the appropriate authentication.

---

## 6 Transport Protocols

### 6.1 Introduction

In order to be able to talk to the external world, the MHP has to be able to communicate through different network types. This part of the MHP specification deals with the Network Independent Protocols and on the networks as defined in two specifications from the DVB project, as specified in [ETS 300 802 \[16\]](#) and [EN 301 192 \[5\]](#).

The protocols defined in these standards provide a generic solution for a variety of broadcast only and interactive services, through the use of DSM-CC User-to-User, Data and Object Carousel protocols, as specified in [ISO/IEC 13818-6 \[26\]](#) and support for IP over the interaction channel as well as over the broadcast channel through the Multiprotocol Encapsulation [EN 301 192 \[5\]](#).

Broadcast only services are provided on systems consisting of a downstream channel from the Service Providers to Service consumers.

Interactive services are provided on systems consisting of a downstream channel together with interaction channels.

There are many possible network configurations covering the currently specified DVB broadcast options including satellite, terrestrial, cable, SMATV and MMDS in conjunction with PSTN, ISDN, cable and other interactive channel options.

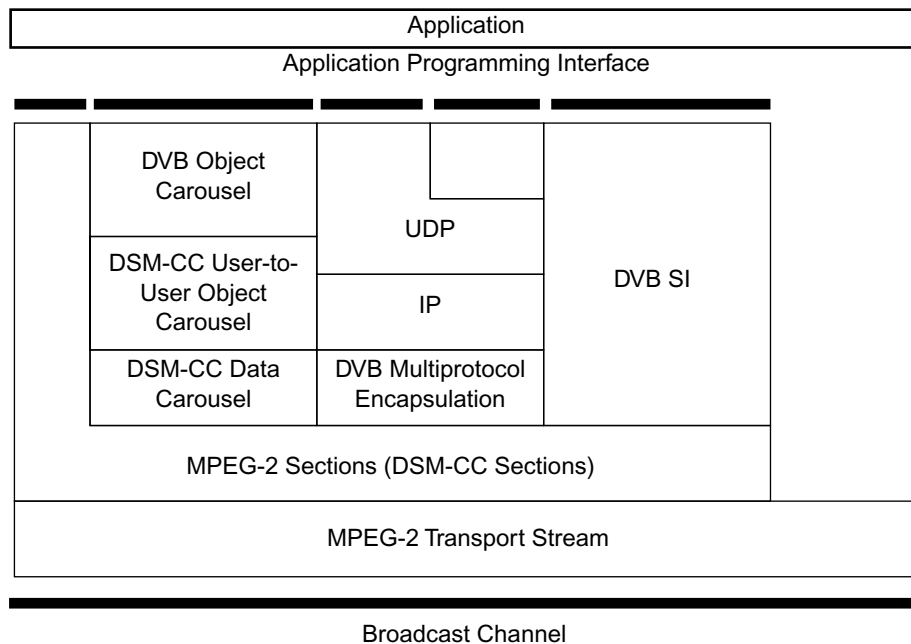
The network dependent protocols for the interaction channels in the DVB context are specified in [ETS 300 800 \[14\]](#), [ETS 300 801 \[15\]](#), [EN 301 193 \[6\]](#), [EN 301 195 \[7\]](#), [EN 301 199 \[8\]](#), [TR 101 201 \[48\]](#) respectively for CATV, PSTN/ISDN, DECT, GSM, LMDS and SMATV networks. The network dependent protocols work together with the Network Independent Protocols.

### 6.2 Broadcast Channel Protocols

This section deals with the DVB defined or referenced broadcast channel protocols. This chapter does not consider other protocols and the APIs that would provide access to them.

Other protocols and their APIs are considered as extensions to the DVB MHP platform, see [K, "\(informative\): Extensions" on page 476](#).

Figure 8 illustrates the set of DVB defined broadcast protocols that are accessible by MHP applications in some or all profiles (see 15, "Detailed platform profile definitions" on page 361). The full details of the APIs that provide access to these broadcast protocols are in chapter 11, "DVB-J Platform" on page 241.



**Figure 8 : Broadcast Channel Protocol Stack**

## 6.2.1 MPEG-2 Transport Stream

MPEG-2 Transport Stream is defined in [ISO/IEC 13818-1 \[23\]](#).

## 6.2.2 MPEG-2 Sections

MPEG-2 private sections as defined in [ISO/IEC 13818-1 \[23\]](#) is [3] is based on the MPEG-2 Transport Stream protocol in 6.2.1.

## 6.2.3 DSM-CC Private Data

DSM-CC Private Data protocol as defined in [ISO/IEC 13818-6 \[26\]](#).

## 6.2.4 DSM-CC Data Carousel

DSM-CC Data Carousel as defined in [ISO/IEC 13818-6 \[26\]](#).

## 6.2.5 DSM-CC User-to-User Object Carousel

DSM-CC User-to-User Object Carousel protocols as defined in [ISO/IEC 13818-6 \[26\]](#) with the restrictions and extensions as defined in [EN 301 192 \[5\]](#), [TR 101 202 \[49\]](#) and annex B, "(normative): Object carousel" on page 388.

### 6.2.5.1 DVB-J class files

Java bytecode for each Java class is carried as the content bytes of the `BIOP::FileMessage` corresponding exactly to the contents of a "class" file as specified in [Java VM \[34\]](#).

### 6.2.5.2 DVB-HTML document files

The set of documents defining a DVB-HTML application is transported with the content bytes of `BIOP::FileMessage` messages corresponding exactly to the contents of the documents (i.e. the `BIOP::FileMessage` doesn't include any HTTP headers, etc.).

### 6.2.5.3 Loss of Carousel Behaviour

Under some conditions, carousels may be lost from the visibility of an application. These conditions are defined in:

- [B.2.9, "Unavailability of a carousel" on page 413](#)
- [9.1.5, "Persistence of Applications Across Service Boundaries" on page 185](#)

When this happens, implementations may continue to provide data from carousels which have been lost where they have that data cached. The extent of this is clearly implementation dependent. It is also implementation dependent how this changes with time. Clearly if data from the lost file system is flushed from a cache, it cannot be replaced. Lost carousels shall never be restored automatically.

Data not in such a cache shall be unavailable to applications. When applications attempt to access data from lost carousels, this shall fail. The failure mode shall be one appropriate to the content format and the mechanism being used to access the data.

Failure modes for DVB-J applications are defined in [11.5.1.3, "Behaviour following loss of a broadcast file system" on page 257](#).

### 6.2.6 DVB Multiprotocol Encapsulation

DVB Multiprotocol Encapsulation as defined in [EN 301 192 \[5\]](#) provides support for IP and is based on the DSM-CC Private Data protocol.

### 6.2.7 Internet Protocol (IP)

Internet Protocol as defined in [RFC 791 \[43\]](#).

### 6.2.8 User Datagram Protocol (UDP)

User Datagram Protocol as defined in [RFC 768 \[42\]](#).

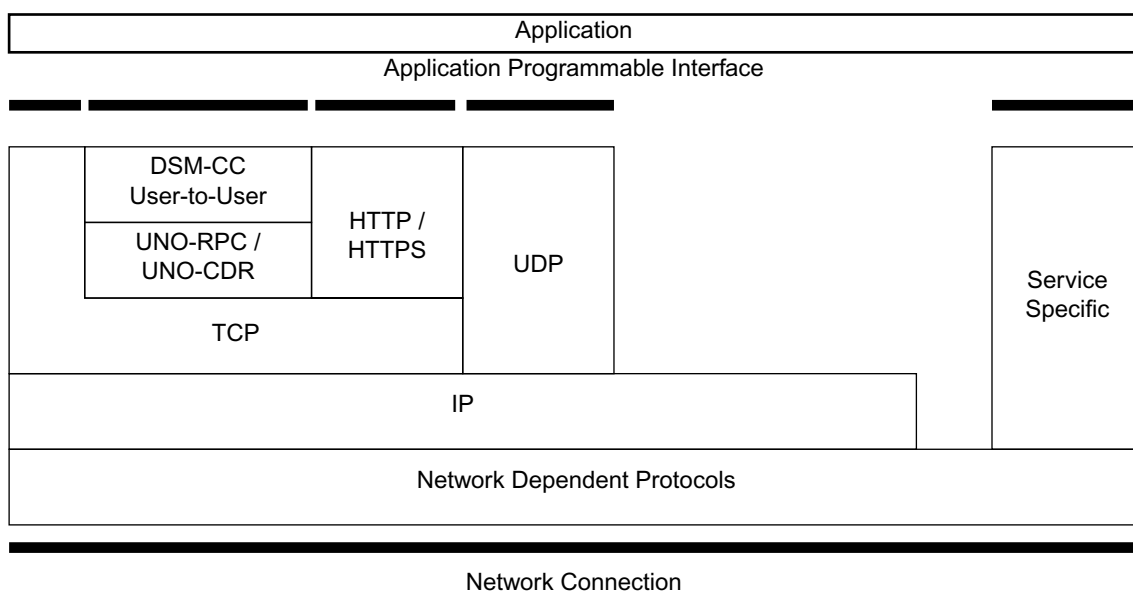
### 6.2.9 DVB Service Information

DVB Service Information as defined in [EN 300 468 \[4\]](#) and [ETR 211 \[11\]](#).

## 6.3 Interaction Channel Protocols

This section deals with the DVB defined or referenced interaction channel protocols. This chapter does not consider other protocols and the APIs that would provide access to them. Other private protocols and possibly APIs are not precluded and are outside of the scope of the MHP.

Figure 9 illustrates the set of DVB defined interaction channel protocols that are accessible by MHP applications in some or all profiles (see 15, "Detailed platform profile definitions" on page 361). The full details of the APIs that provide access to these interaction protocols are in chapter 11, "DVB-J Platform" on page 241.



**Figure 9 : Interaction Channel Protocol Stack**

### 6.3.1 Network Dependent Protocols

As defined in ETS 300 800 [14], ETS 300 801 [15], EN 301 193 [6], EN 301 195 [7], EN 301 199 [8], TR 101 201 [48] respectively for CATV, PSTN/ISDN, DECT, GSM, LMDS and SMATV networks.

### 6.3.2 Internet Protocol (IP)

Internet Protocol as defined in RFC 791 [43].

### 6.3.3 Transmission Control Protocol (TCP)

Transmission Control Protocol as defined in RFC 793 [44].

### 6.3.4 UNO-RPC

The UNO-RPC consists of the Internet Inter-ORB Protocol (IIOP) as specified in CORBA/IIOP [2].

### 6.3.5 UNO-CDR

The UNO-CDR as defined in CORBA/IIOP [2].

### 6.3.6 DCM-CC User to User

DSM-CC User-to-user as defined in ISO/IEC 13818-6 [26] with the restrictions and extensions as defined in EN 301 192 [5] and TR 101 202 [49].

## 6.3.7 Hypertext Transfer Protocol (HTTP)

### 6.3.7.1 HTTP 1.1

Hypertext Transfer Protocol as defined in [RFC 2616 \[40\]](#).

### 6.3.7.2 MHP profile of HTTP 1.0

HTTP 1.0 is defined in [RFC 1945 \[78\]](#). MHP terminals supporting HTTP 1.0 are required to support persistent connections as defined below. MHP terminals implementing profiles where HTTP 1.0 is required are also allowed to implement subsequent versions of the HTTP specification (e.g. HTTP 1.1, [RFC 2616 \[40\]](#)) as long as these are backwards compatible and persistent connections or their equivalent in the subsequent version of the specification are supported.

With no support for persistent connections, a separate TCP connection has to be established to fetch each URL, increasing the load on HTTP servers and causing congestion. The use of multiple separate image files and other associated data, for example often require a client to make multiple requests to the same server in a short amount of time. Therefore an MHP terminal implementation shall include support for persistent connections.

The following section explains how persistent connection is to be used within the MHP. The original form of HTTP 1.0 persistent connections is defined as informational text in [RFC 2068 \[J\]](#). The normative specification text included here is intended to be fully compatible with this RFC.

#### 6.3.7.2.1 HTTP 1.0 persistent connections

When connecting to a server, an MHP terminal shall send the Keep-Alive connection-token:

```
Connection: Keep-Alive
```

An HTTP server (1.0 or 1.1) would then respond with the Keep-Alive connection token and the client may proceed with an HTTP 1.0 (Keep-Alive) persistent connection.

#### 6.3.7.2.2 The Keep-Alive Header

When the Keep-Alive connection-token has been transmitted with a request or a response, a Keep-Alive header field may also be included. The Keep-Alive header field takes the following form:

```
Keep-Alive-header = "Keep-Alive" ":" 0# keepalive-param
keepalive-param = param-name "=" value
```

where the "0#" notation means that the "keepalive-param" field may be repeated 0 or more times and they are separated by a comma character "," if the field is included more than once.

The Keep-Alive header itself is optional, and is used only if a parameter is being sent. This specification does not define any parameters.

If the Keep-Alive header is sent, the corresponding connection token must be transmitted. The Keep-Alive header must be ignored if received without the connection token.

#### 6.3.7.2.3 MHP and proxies

An MHP terminal must not send the Keep-Alive connection token to a HTTP 1.0 proxy server, as HTTP 1.0 proxy servers do not obey the rules of HTTP 1.1 for parsing the Connection header field.

If an MHP terminal sends Keep-Alive to a proxy server that doesn't understand Connection, which would then erroneously forward it to the next inbound server, which would establish the Keep-Alive connection and result in a hung HTTP 1.0 proxy waiting for the close on the response. The result is that the MHP terminal must be prevented from using Keep-Alive when talking to proxies that doesn't understand Connection.

#### 6.3.7.2.4 Version compatibility

An HTTP 1.1 server may also establish persistent connections with an MHP terminal upon receipt of a Keep-Alive connection token. However, a persistent connection with an MHP terminal can't make use of the chunked transfer coding, and therefore MUST use a content-length for marking the ending boundary of each message.

For servers used for MHP applications, it is recommended that if HTTP 1.1 servers are used, they should support the HTTP 1.0 persistent connections when initiated by an HTTP 1.0 client using the Keep-Alive connection token.

### 6.3.7.3 HTTPS

This is described in [RFC 2818 \[98\]](#).

### 6.3.8 Service Specific

Service Specific protocols are proprietary protocols used by a service as defined in the guidelines for the Data Broadcast Specification [TR 101 202 \[49\]](#). The DVB provides a registry mechanism for new, proprietary broadcast protocols.

### 6.3.9 User Datagram Protocol (UDP)

User Datagram Protocol as defined in [RFC 768 \[42\]](#).

## 6.4 Transport protocols for application loading over the interaction channel

Three scenarios are envisaged:

- The file system is completely implemented in the broadcast channel (the classic MHP 1.0 model which is not described further here)
- File system implemented only via the interaction channel
- Hybrid between broadcast stream and interaction channel

### 6.4.1 File system implemented only via the interaction channel

This clause addresses the case where the interaction channel presents the only file system i.e. where the protocol ID is 0x0003.

#### 6.4.1.1 File system logical structure

The list of items signalled in the transport protocol descriptor(s) (see 10.8.1.3, "Transport via interaction channel" on page 226) are considered to form a single name space.

The MHP terminal when trying to locate a file specified by an incomplete (relative) filename, shall attempt to fetch a file from each of the items in that list in the order in which they are found in the list until the file is found or the list is exhausted.

The items in the list shall either be references to .zip files (see ZIP [77]) or base URLs ending in '/' onto which the path of the requested file shall be concatenated. Any items in the list which are not one of these two types shall be ignored. Errors in discovering whether a specific file can be reached through a specific list item shall be ignored. The list must contain at least one item.

As an example, consider the MHP platform retrieving the "dvb.fontindex" file for an application from an which AIT includes two transport protocol descriptors with the following contents;

**Table 1 : Example transport descriptor selector byte payload**

URL base	URL extension
"http://www.dvb.org"	"applications/application1.zip"
	"graphics/"
	"shared/utills.zip"
"http://www.ebu.ch"	"general/misc.zip"
"http://www.dvb.org"	"other_stuff/we_dont_use_this_very_often.zip"

Then the search order for "dvb.fontindex" would be the following:-

- a) in the contents of http://www.dvb.org/applications/application1.zip
- b) from the URL http://www.dvb.org/graphics/dvb.fontindex
- c) in the contents of http://www.dvb.org/shared/utills.zip
- d) in the contents of http://www.ebu.ch/general/misc.zip
- e) in the contents of http://www.dvb.org/we\_dont\_use\_this\_very\_often.zip

The search order for "abc/dvb.hashfile" would be the following:-

- a) for "abc/dvb.hashfile" in the contents of http://www.dvb.org/applications/application1.zip
- b) from the URL http://www.dvb.org/graphics/abc/dvb.hashfile
- c) for "abc/dvb.hashfile" in the contents of http://www.dvb.org/shared/utills.zip
- d) for "abc/dvb.hashfile" in the contents of http://www.ebu.ch/general/misc.zip



These file system semantics shall be followed for all files fetched by or on behalf of an MHP application. This includes the files whose names and locations are standardised by this specification, e.g. 'dvb.fontindex', 'dvb.hashfile', 'dvb.signature.x' and 'dvb.certificate.x'.

For DVB-J, with the following application location descriptor:-

- a) base\_directory "/app1/starting"
- b) classpath\_extension "/shared;/app1/later"
- c) initial\_class "org.dvb.demo.myXlet"

The search order for a class would be the following:-

- a) for "app1/starting/org/dvb/demo/myXlet.class" in the contents of <http://www.dvb.org/applications/application1.zip>
- b) from the URL <http://www.dvb.org/graphics/app1/starting/org/dvb/demo/myXlet.class>
- c) for "app1/starting/org/dvb/demo/myXlet.class" in the contents of <http://www.dvb.org/shared/utills.zip>
- d) for "app1/starting/org/dvb/demo/myXlet.class" in the contents of <http://www.ebu.ch/general/misc.zip>
- e) other\_stuff/we\_dont\_use\_this\_very\_often.zip
- f) for "/app1/starting/shared/org/dvb/demo/myXlet.class" in the contents of <http://www.dvb.org/applications/application1.zip>
- g) from the URL <http://www.dvb.org/graphics/app1/starting/shared/org/dvb/demo/myXlet.class>
- h) for "/app1/starting/shared/org/dvb/demo/myXlet.class" in the contents of <http://www.dvb.org/shared/utills.zip>
- i) for "/app1/starting/shared/org/dvb/demo/myXlet.class" in the contents of <http://www.ebu.ch/general/misc.zip>
- j) other\_stuff/we\_dont\_use\_this\_very\_often.zip
- k) for "/app1/starting/app1/later/org/dvb/demo/myXlet.class" in the contents of <http://www.dvb.org/applications/application1.zip>
- l) from the URL <http://www.dvb.org/graphics/app1/starting/app1/later/org/dvb/demo/myXlet.class>
- m) for "/app1/starting/app1/later/org/dvb/demo/myXlet.class" in the contents of <http://www.dvb.org/shared/utills.zip>
- n) for "/app1/starting/app1/later/org/dvb/demo/myXlet.class" in the contents of <http://www.ebu.ch/general/misc.zip>
- o) other\_stuff/we\_dont\_use\_this\_very\_often.zip

#### 6.4.1.2 File transfer

Files shall be transferred using the profile of HTTP specified under 6.3.7.2, "MHP profile of HTTP 1.0" on page 58.

#### 6.4.1.3 Class encoding

The classes of DVB-J applications can be delivered either as discrete class files or within ZIP files (see ZIP [77]).

NOTE: This contrasts to the broadcast case where object carousel files are simply class files.

The classes of DVB-J applications shall be delivered as discrete class files at the level of the file system.

NOTE: This means that DVB-J class files are allowed to be held on an HTTP server either as discrete class files or in ZIP files (see ZIP [77]) where ZIP files form part of the overall file system.

#### 6.4.1.4 Directory listing in this file system

This file system is considered to be one that does not support directory listing in the following contexts:

- authentication (see 12.4.1.4, "Special authentication rules" on page 290).
- listing the files in a directory (see 14.7, "Files and file names" on page 358).

## 6.4.2 Hybrid between broadcast stream and interaction channel

In the hybrid case all directory information is provided in the broadcast stream but some, or possibly all, of the file contents are provided via the interaction channel.

### 6.4.2.1 File transfer

#### 6.4.2.1.1 Broadcast file delivery

The file contents are carried via a BIOP::File as is the normal case for an Object Carousel defined in MHP 1.0. As described in MHP 1.0 the IOR from the file binding to the file contents uses either a BIOPProfileBody or a LiteOptionsProfileBody.

#### 6.4.2.1.2 Interaction channel delivery

In this case the IOR from the file binding to the file contents uses the HTTPProfileBody (see table 2) to identify the location of the file contents on the interaction channel.

This form of IOR shall only be used for BIOP::File objects.

See 6.3.7.2, "MHP profile of HTTP 1.0" on page 58 for the profile of HTTP that is used to retrieve the contents of the file.

#### 6.4.2.1.3 HTTPProfileBody

The HTTP Profile body specifies host, port and path\_segments. In HTTP requests these are concatenated to form a URL of the form:

```
http://host:port/path_segments
```

**Table 2 : HTTP Profile Body syntax**

Syntax	bits	Type	Value	Comment
HTTPProfileBody {				
profileId_tag	32	uimsbf	0x44564200	
profile_data_length	32	uimsbf	*	
profile_data_byte_order	8	uimsbf	0x00	big endian byte order
version.major	8	uimsbf	0x01	protocol major version 1
version.minor	8	uimsbf	0x00	protocol minor version 0
host_data_length	8	uimsbf	N1	
for (k=0; k<N1; k++) {				
host_data	8	uimsbf	+	
}				
port	16	uimsbf		
objectKey_length	16	uimsbf	N2	<= 4
for (k=0; k<N2; k++) {				
objectKey_data	8	uimsbf	+	
}				
}				

NOTE: The tag values 0x44564201...0x44564201 have been reserved for future use by the DVB. 0x44 is ASCII for "D", 0x56 is "V" and 0x42 is "B".

**version:** These 8-bit fields indicate the version of the protocol that the server will use to deliver the file specified. The version value 1.0 indicates that the transport protocol is as defined in 6.3.7.2, "MHP profile of HTTP 1.0" on page 58.

**host\_data:** These bytes convey the identifier of the Internet host to which messages may be sent. It may be a fully qualified domain name or Internet standard "dotted decimal" form (e.g., "192.231.79.52").

**port:** This 16 bit unsigned integer is the TCP/IP port number at the specified host where the target agent is listening for requests.

**objectKey\_data:** These bytes form a string which carries the path\_segments portion of the URL which uniquely identifies the object on the server [RFC 2396 \[41\]](#).

#### 6.4.2.2 Class encoding

Each class of the DVB-J application shall be delivered in an individual file regardless of whether the file contents are retrieved from the broadcast channel or from the interaction channel.

## 7 Content formats

### 7.1 Static formats

#### 7.1.1 Bitmap image formats

##### 7.1.1.1 Image encoding restrictions

Any indications in the transmitted image with respect to pixel scaling, colour space or gamma are to be ignored in the presenting of the image. One image pixel shall be mapped to one graphics pixel in the current graphics configuration, unless otherwise scaled by the application directly.

See also 7.5, "Colour Representation" on page 67.

##### 7.1.1.2 JPEG

JPEG as defined in ISO/IEC 10918-1 [21] using the JFIF [35] file exchange format.

Only coding using sequential DCT-based mode or progressive DCT-based mode is required to be supported by implementations.

Specifically, lossless and hierarchical modes need not be supported.

##### 7.1.1.3 PNG

PNG is defined as in PNG [37].

See also 15.1, "PNG - restrictions" on page 363.

##### 7.1.1.4 GIF

GIF is defined as in GIF 89a [17].

#### 7.1.2 MPEG-2 I-Frames

MPEG-2 I-Frames are defined as in ISO/IEC 13818-2 [24].

The payload of a file delivering an MPEG -2 I frame shall:

- be a valid video\_sequence() including a sequence\_extension()
- contain one I frame only, i.e. one picture\_header(), one picture\_coding\_extension(), and one picture\_data() encoded as an intra coded frame, with picture structure = "frame"

That is the structure is:

```
sequence_header()
sequence_extension()
extension_and_user_data(0)
optional group_of_pictures_header() and extension_and_user_data(1)
picture_header ( picture_coding_type = "I frame")
picture_coding_extension ( picture_structure = "frame picture")
extension_and_user_data(2)
picture_data()
sequence_end_code()
```

#### 7.1.3 MPEG-2 Video "drips"

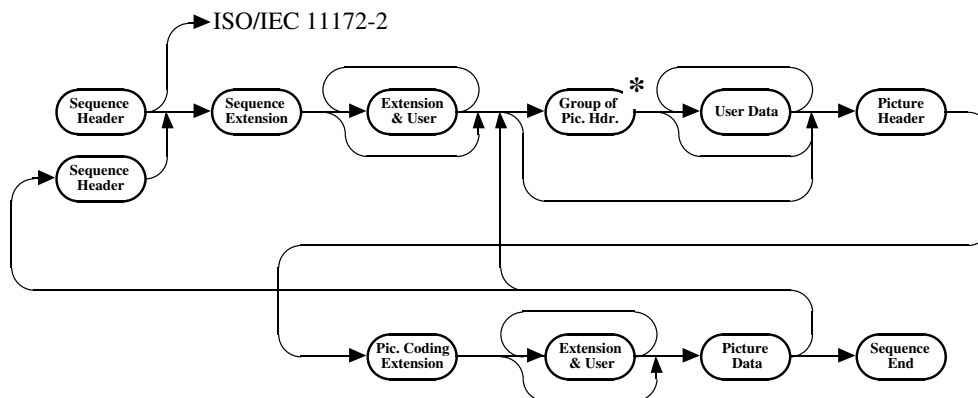
The drip feed mode consists of letting an application progressively feed the MPEG-2 video decoder with chunks of an MPEG-2 video stream. In this mode, it is only required for the decoder to handle I and P frames (i.e. not B frame). Each chunk shall contain one frame and a certain number of syntactic elements (as described in ISO/IEC 13818-2 [24]) such as sequence\_header() or group\_of\_picture\_header().

Firstly, the content of each of the chunks of bytes fed to the decoder shall comply with the following syntax:

```
optional {
  sequence_header()
  sequence_extension()
  extension_and_user_data(0)
  optional {
    group_of_pictures_header()
    extension_and_user_data(1)
  }
}

picture_header ( picture_coding_type = "I frame" or "P frame")
picture_coding_extension ( picture_structure = "frame picture")
extension_and_user_data(2)
picture_data()
optional {
  sequence_end_code()
}
```

In addition, the overall concatenation of chunks over time shall respect the authorized combinations of syntactic elements described in ISO/IEC 13818-2 [24] to build a legal MPEG-2 video stream. The following diagram, extracted from ISO/IEC 13818-2 [24], reflect the rules defined in that standard:



\* After a GOP the first picture shall be an I-picture

**Figure 10**

The following restrictions are applied to P-frames:

- The P-frame shall contain no prediction information (i.e. no motion vector shall be present in macroblock elements).
- The allowed macroblock\_types for P-frames are:
  - "Intra" (i.e. VLC code 0001 1)
  - "Intra, Quant" (i.e. VLC code 0000 01)
  - "No MC, Coded" (i.e. VLC code 01)
  - "No MC, Coded, Quant" (i.e. VLC code 0000 1)

NOTE: The standard semantics for P-frames allow macroblock\_escape and macroblock\_address\_increment to signal skipped macroblocks. This allows P-frames to be very sparse, only carrying macroblocks positioned at certain locations on the screen. This contrasts with semantics for an I-frame where macroblocks are required to fill the full screen.

If invalid content is fed to the MPEG-2 video decoder, the content is discarded and there are no guarantees when subsequent valid chunk of byte fed to the decoder will be displayed (unless the decoder is restarted).

This mode requires the decoder to be in the "low delay" mode as defined in ISO/IEC 13818-2 [24].

This mode can be used by connecting a `org.dvb.media.DripFeedDataSource` instance to a Player representing a MPEG-2 video decoder. See [N, "\(normative\): Streamed Media API Extensions" on page 574](#).

## 7.1.4 Monomedia format for audio clips

The format for audio clips is MPEG-1 Audio (Layer 1 & 2) ES data as defined as in [ISO/IEC 11172-3 \[22\]](#) and constrained in [TR 101 154 \[9\]](#).

Each "file" of audio content is a binary data file carrying Audio elementary stream data. Each "file" delivers an integer number of audio access units and the first byte of each file is the first byte of an audio access unit. The MPEG Audio data in all other respects conforms to the specifications provided in [TR 101 154 \[9\]](#).

Implementations decoding audio clips can assume that they have an approximately constant number of bytes per second. If this not true then the behaviour is implementation dependent.

## 7.1.5 Monomedia format for text

Java modified UTF-8 as defined in [Java Language Spec \[32\]](#) section 22.2.14 "writeUTF" is the coding of text in MHP.

NOTE: Based on [ISO 10646-1 \[18\]](#) but modified with respect to the encoding of the character code zero.

### 7.1.5.1 Built-in character set

See [E, "\(normative\): Character set" on page 434](#).

## 7.2 Broadcast streaming formats

### 7.2.1 Audio

MPEG Audio with the restrictions and enhancements defined in [TR 101 154 \[9\]](#)

### 7.2.2 Video

Standard Definition 25 Hz MPEG Video with the restrictions and enhancements defined in [TR 101 154 \[9\]](#).

### 7.2.3 Subtitles

The content formats supported for subtitles are:

- DVB Subtitles
- Teletext

See [13.5, "Subtitles" on page 349](#).

In the event that both DVB Subtitles and DVB Teletext are available then DVB Subtitles will take precedence (i.e. if a stream is flagged as having both DVB Subtitles and Teletext Subtitles then the DVB Subtitles will be displayed).

Teletext Subtitles conform to the same display model, as DVB subtitles.

Application control and detection of subtitles, whether they be DVB Subtitles or Teletext Subtitles, will be through JMF. The application will have no knowledge of the delivery/presentation protocol being used to provide subtitles.

No APIs will be provided to access Teletext data packets and no timing model is provided for the decoding of Teletext subtitle. Text subtitles will be decoded as soon as the data becomes available.

#### 7.2.3.1 DVB Subtitles

DVB Subtitles are defined as in [EN 300 743 \[13\]](#).

### 7.2.3.2 Teletext

Transmission of the text is as defined in EN 300 472 [12]. The data format is as defined in ETS 300 706 [61] but restricted to presentation level 1.5 or lower. Signalling of the Teletext subtitle page will be via the Teletext descriptor as defined in EN 300 468 [4].

Within the MHP specification Teletext is only supported as an alternative content format for delivery of subtitles. The MHP specification does not address its possible use as a navigable content format.

NOTE: Manufactures remain free to implement full Teletext support based on regulatory requirement or market demand. Such support would be implemented outside of the MHP environment, by VBI re-insertion of the non-subtitle text or through a native Teletext Decoder. The user interface integration is then an issue for the manufacturer to resolve.

It is envisaged that broadcasters will use MHP applications to deliver navigable text services providing a greater level of interactivity and enhanced graphics.

## 7.3 Resident fonts

See section G.4, "Resident fonts and text rendering" on page 445.

See also annex D, "(normative): Text presentation" on page 421.

## 7.4 Downloadable Fonts

PFR0 (Portable Font Resource version 0) is defined as in DAVIC 1.4.1p9 [3] as the coding format for fonts. Receivers are only required to provide support for the outline version of the font.

The `charCode` value in the PFR `charRecord` shall be the ISO 10646-1 [18] code for the glyph encoded using UCS-2.

See also D.2.2, "Downloaded fonts" on page 421.

## 7.5 Colour Representation

### 7.5.1 Background (informative)

The method of colour encoding is critical to how consistently the colours in an image can be reproduced across different systems. The description must be cast in a way which is independent of the mechanisms by which it will finally be reproduced for the viewer.

The International Colour Consortium (ICC) has proposed a thorough solution to the precise communication of colour in open systems. However the ICC profile format is somewhat over-specified for the MHP. The ICC mechanism for ensuring that a colour is correctly mapped from an input to the output colour space is by attaching a profile for the input colour space to the image in question. This is appropriate for high end systems, especially those in the print media. However, a primarily CRT based home platform neither needs, nor has the processing power and available bandwidth, to handle an embedded profile mechanism. It would also require some sophistication on the part of the end consumer to set up properly.

Fortunately by adopting a single default colour space that can be processed as an **implicit** ICC profile the advantages of the ICC approach are gained, and the system is later scalable to a full colour management system with a clear relationship to existing ICC colour management systems while minimizing software and support requirements in an MHP today.

A colour space is a model for representing colour numerically in terms of three or more coordinates or tristimulus values. An RGB colour space represents colours in terms of Red, Green and Blue coordinates. The MHP format shall use the specific RGB encoding for colour imagery, sRGB as defined in IEC 61966-2-1 [27]. This is suitable for a wide range of presentation environments including TV's and has become widely adopted in the computer environment and WWW. It is, for example, compatible with CCIR Recommendation ITU-R BT.709 [30] standard for colour encoding in HDTV. This format has the advantage of device independence without a great deal of additional overhead.

For sRGB, the goal is to communicate the appearance of colours as displayed on a reference monitor in terms of 8-bit digital code values for each coordinate. sRGB colour values represent colour appearance with respect to a defined reference viewing environment.

For colour stimuli viewed in the reference viewing environment, sRGB values are defined by a series of simple mathematical operations from standard CIE colourimetric values.

The sRGB format is a good match for 24 bit colour on most CRT's. In devices where a great deal of damage is done to the colour space it may not give consistent results. For example dithering to a 4-bit per primary colour map will violate the gamma assumptions.

## 7.5.2 Specification

All images transmitted shall be within the gamut encompassed by the sRGB colourspace. Where possible this should be coded so that the terminal does not have to translate. Where this is impractical the sRGB image may be transcoded into a different colourspace provided the gamut assumption is not violated (i.e. to be consistent with JFIF, JPEG images shall be sent in the region of the  $Y_C C_b$  colourspace that overlaps with the sRGB gamut).

NOTE: that the presentation of images using colours outside of the sRGB gamut shall be platform dependent.

Images created in the MHP will be in the sRGB colourspace by default, although manufacturers are free to provide support for other colour spaces if they choose. All MHPs shall support transformations from sRGB to the colour spaces allowed by the MPEG-2 definition (e.g. BT 709 and BT 420) and vice versa, manufacturers may choose to support transformations to and from other colour spaces.

### 7.5.2.1 The sRGB Reference Viewing Environment

The reference display conditions and viewing environment for sRGB are partly described in table 3. A reference viewing environment must be provided to allow for the unambiguous definition of colour, the sRGB reference viewing environment corresponds to conditions typical of indoor viewing of CRTs – further details can be found in the IEC 61966-2-1 [27].

The sRGB reference conditions therefore provides a well defined reference compatible with ITU-R BT.709 [30].

**Table 3 : sRGB reference Display conditions**

Condition	sRGB
Viewing flare	1,0 %
Reference Background	20 %
Display model Offset	0,055
Display Gun/Phosphor Gamma	2,4
Display white point	$x = 0,3127$ $y = 0,3290$ (D65 Hunt, R.W.G. [52])
Ambient Lighting	64 lx
Display Luminance level	80 cd/m <sup>2</sup>

### 7.5.2.2 Colourimetric Definitions and Encodings

sRGB tristimulus values can be computed as follows, firstly linear sRGB tristimulus are computed as linear combinations of the 1931 CIE XYZ (CIE 15 [1]) values using the following relationship:

$$\begin{bmatrix} R_{sRGB} \\ G_{sRGB} \\ B_{sRGB} \end{bmatrix} = \begin{bmatrix} 3,2410 & -1,5374 & -0,4986 \\ -0,9682 & 1,8760 & -0,0416 \\ 0,0556 & -0,2040 & -1,0570 \end{bmatrix} \begin{bmatrix} X_{D65} \\ Y_{D65} \\ Z_{D65} \end{bmatrix}$$



In the encoding process, negative sRGB tristimulus values, and sRGB tristimulus values greater than 1,00 are not retained. The luminance dynamic range and colour gamut of sRGB is limited to the tristimulus values between 0,0 and 1,0 by simple clipping. This gamut, however, is large enough to encompass most colours that can be displayed on CRT monitors.

For comparison, the CIE chromaticities for the red, green, and blue ITU-R BT.709 and ITU-R BT.470 reference primaries, and for CIE Standard Illuminant D65 (IEC 61966-2-1 [27]), are given in tables 4, 5. From these primaries the  $Y C_b C_r$  transmitted values are computed by similar relationships.

Therefore ITU-R BT.709  $Y C_b C_r$  colour space and similar video colour spaces can be converted to sRGB and vice versa by way of CIE XYZ. Chromaticities for other video formats allowed in MPEG streams can be found in their respective standards.

**Table 4 : ITU-R BT.709 reference primaries and CIE standard illuminant**

	Red	Green	Blue	D65
<b>x</b>	0,6400	0,3000	0,1500	0,3127
<b>y</b>	0,3300	0,6000	0,0600	0,3290
<b>z</b>	0,0300	0,1000	0,7900	0,3583

**Table 5 : TU-R BT.470-2 reference primaries and CIE standard illuminant**

	Red	Green	Blue	D65
<b>x</b>	0,6700	0,2100	0,1400	0,3100
<b>y</b>	0,3300	0,7100	0,0800	0,3160
<b>z</b>	0,0100	0,0800	0,7800	0,3740

The linear sRGB tristimulus values are next transformed to non linear sR'G'B' values. This process closely approximates the effect of a "gamma" curve of 2,2 with a slight offset. This makes sRGB consistent with legacy systems and images.

if  $R_{sRGB} > 0,00304$  and  $G_{sRGB} > 0,00304$  and  $B_{sRGB} > 0,00304$

then

$$R'_{sRGB} = 1,055 * R_{sRGB}^{(1,0/2,4)} - 0,055$$

$$G'_{sRGB} = 1,055 * G_{sRGB}^{(1,0/2,4)} - 0,055$$

$$B'_{sRGB} = 1,055 * B_{sRGB}^{(1,0/2,4)} - 0,055$$

else

$$R'_{sRGB} = 12,92 * R_{sRGB}$$

$$G'_{sRGB} = 12,92 * G_{sRGB}$$

$$B'_{sRGB} = 12,92 * B_{sRGB}$$

end if

Finally, the non-linear sR'G'B' values are converted to digital code values. This conversion scales the sR'G'B' values by using the equation below where WDC represents the white digital count and KDC represents the black digital count.

$$R_{8bit} = ((WDC - KDC) * R'_{sRGB}) + KDC$$

$$G_{8bit} = ((WDC - KDC) * G'_{sRGB}) + KDC$$

$$B_{8bit} = ((WDC - KDC) * B'_{sRGB}) + KDC$$

The current sRGB specification uses a black digital count of 0 and a white digital count of 255 for 24-bit (8-bits/channel) encoding, and the MHP shall adopt the same convention. However, note that some digital video signals may use a black digital count of 16 and a white digital count of 235 in order to provide a larger encoded colour gamut.

Details of the reverse transformation from sRGB to CIE XYZ are given in IEC 61966-2-1 [27], mappings from ITU-R BT.709 and ITU-R BT.470 to CIE XYZ are given in ISO/IEC 13818-2 [24].

## 7.6 MIME Types

**Table 6 : File type identification**

MIME type	Extension (note 1)	Definition of content
"image/jpeg"	".jpg"	As defined in 7.1.1.2, "JPEG" on page 64.
"image/png"	".png"	As defined in 7.1.1.3, "PNG" on page 64 possibly with a constrained profile as defined in 15.1, "PNG - restrictions" on page 363.
"image/gif"	".gif"	As defined in 7.1.1.4, "GIF" on page 64.
"image/mpeg"	".mpg"	As defined in 7.1.2, "MPEG-2 I-Frames" on page 64.
"video/mpeg"	".mpg"	As defined in 7.2.2, "Video" on page 66.
"video/dvb.mpeg.drip"	".drip"	As defined in 7.1.3, "MPEG-2 Video "drips"" on page 64.
"audio/mpeg"	".mp2"	As defined in 7.1.4, "Monomedia format for audio clips" on page 66 or as defined in 7.2.1, "Audio" on page 66.
"text/dvb.utf8"	".txt"	As defined in 7.1.5, "Monomedia format for text" on page 66.
"text/xml"	".xml"	As defined in 8, "DVB-HTML" on page 71.
"text/css"	".css"	As defined in 8.8, "CSS" on page 103.
"text/plain"	".txt"	As defined in 7.1.5, "Monomedia format for text" on page 66.
"text/ecmascript"	".js"	As defined in 8.6.11, "Script content" on page 89.
"image/dvb.subtitle"	".sub"	As defined in 7.2.3, "Subtitles" on page 66.
"text/dvb.subtitle"		
"text/dvb.teletext"	".tlx"	As defined in 7.2.3.2, "Teletext" on page 67.
"application/dvb.pfr"	".pfr"	As defined in 7.4, "Downloadable Fonts" on page 67.
"application/dvbj"	".class"	A DVB-J class file. See 6.2.5.1, "DVB-J class files" on page 55.
"application/xml"	".xml"	As defined in 8, "DVB-HTML" on page 71.
"multipart/dvb.service"	".svc"	An MPEG Program (DVB Service) conforming to DVB norms.
NOTE 1: Future formats may use more characters in the extension		

### 7.6.1 Rationale

The MIME types are defined to reserve a name space for the possible future support of downloadable JMF players.

The file name extensions shall be included in broadcasts to assist receivers identify the type of the content. For DVB-J applications, this is described in table 113, "Return types of URL.getContent()" on page 245).

Not all MIME and filename extensions defined in the above table are actually used in this specification. For DVB-J, the APIs which consume media types are described in 15.2, "Minimum media formats supported by DVB-J APIs" on page 364. With MIME types whose corresponding media is not listed for a particular profile, access to that content type from files is not defined for that profile.

---

## 8 DVB-HTML

### 8.1 Introduction

#### 8.1.1 Application Area

One application area considered of commercial importance for the Multimedia Home Platform (MHP) is that of the provision of a hypertext or "super teletext" system for the presentation of information alongside broadcast. In order to promote interoperability, both between different MHP vendors and where possible with the wider internet, this document describes a specific subset of the data formats defined for use in the World Wide Web adopted by the MHP, plus DVB specific extensions.

The DVB MHP specification provides the basic definitions needed for integration of DVB HTML applications into the MHP:

- Definition of the term DVB HTML application and its lifecycle in 9.3.1, "[The DVB-HTML Application](#)" on page 193.
- How to signal a DVB HTML application in 10, "[Application Signalling](#)" on page 208.
- Extensions on how to transport a DVB HTML application in 6.2.5.2, "[DVB-HTML document files](#)" on page 55.
- The definition of content for DVB HTML in this chapter.

## 8.1.2 Profiles

There is only one DVB-HTML profile defined see (15, "Detailed platform profile definitions" on page 361).

# 8.2 Architecture

## 8.2.1 Context

A DVB-HTML application in the context of MHP is a set of resources selected from the DVB-HTML family of content formats as defined in this specification. The resources of a DVB-HTML application form a directed graph, rooted by the resource referenced in the Application Entry Point (see "DVB-HTML application location descriptor" on page 232). The allowed extent of the graph is described by the application boundary descriptor, the nodes of the graph are the physical representation of the resources, the arcs are naming references defined in the content formats. The interpretation of a DVB-HTML application is handled by a support application (commonly called a user agent in W3C terminology). For the purposes of exposition DVB defines an "actor" which is the runtime context associated with a DVB-HTML application maintained by the user agent, there is a one to one correspondence between actors and running applications. There may be a many to one relationship between actors and user agents. The lifecycle of a DVB-HTML application is described in 9.3, "DVB-HTML Model" on page 193.

During the lifetime of a DVB-HTML application the user agent performs three types of processing on resources to enable the behaviour of the DVB-HTML application:

- **Decoding**
- **Presentation**
- **Interaction**

**Decoding** is the process of reading the bit representation of the resource and constructing a runtime representation in the actor context. The nature of the runtime representation is not specified, but it shall respect the required semantics of the resource type. Most resource types have a visible or audible representation, and **Presentation** is the process of rendering the runtime representation of a resource to the relevant device. Many resources can be partially presented whilst decoding is taking place, it is an implementation option as to whether this occurs in any given user agent. Some resource types have a behavioural model, and **Interaction** is the generic term used for the processing required by the user agent to correctly exhibit the behaviour model defined by the semantics of the resource. Similarly it is possible that the interaction processing may overlap with decoding and presentation.

## 8.2.2 Integration Aspects

### 8.2.2.1 Accessing DVB-J from ECMAScript

All of the APIs available to DVB-J shall be available to ECMAScript, See 8.10.2, "Interface between ECMAScript and DVB-J" on page 122.

### 8.2.2.2 Implementation of user agents via plug-ins

In the case that the user agent is implemented as an interoperable plug-in, at least the following parts of the specification will be relevant:

- 5, "Basic Architecture" on page 48
- 9.9, "Plug-ins" on page 207
- 10, "Application Signalling" on page 208
  - 10.13, "Plug-in signalling" on page 236
- 12, "Security" on page 286
  - 12.13, "Plug-ins" on page 331

## 8.3 Application Format

### 8.3.1 Basic Considerations

The main parts of this specification are based on World Wide Web Consortium (W3C) recommendations. W3C provides the fundamental building block standards of the WWW, that are also used in the DVB-HTML standard as well as for HTML standards for other devices like cell phones, PDAs and Internet appliances. Authoring tools and software used for displaying content for the WWW are based on these recommendations. This specification uses W3C recommendations, as far as possible, to allow authoring of a common content basis for both the DVB-HTML applications and the content distributed on the WWW.

In particular the following technologies are used:

- XHTML (see 8.5 on page 74)
- CSS (see 8.8 on page 103)
- DOM (see 8.11 on page 128)
- ECMAScript (see 8.10 on page 122)
- XML (see 8.4 on page 74)

### 8.3.2 Approach to Subsetting

The W3C recommendations contain conformance clauses under which a product or code implementing the specification is considered a conformant implementation. In this specification this approach is supported to the largest extent possible. If in special cases deviation is required it is clearly marked in the text.

## 8.4 XML

XML is a license-free, platform-independent and well-supported method for structuring data in textual form. The DVB-HTML language as defined in section 8.5, "DVB Mark-up Language (DVB-HTML)" on page 74 conforms to XML 1.0 [65]. Since its creation XML has become the foundation of new internet mark-up languages specified by the W3C.

The grammar of the DVB-HTML language is described in the DTD (Document Type Definition) in annex AA "(normative): DVB-HTML 1.0 DTD". This DTD conforms to the XHTML Host Language Document Type conformance clauses defined in the XHTML modularization specification Modularization of XHTML [80].

## 8.5 DVB Mark-up Language (DVB-HTML)

DVB-HTML is an application of XML. The XML document character set for DVB-HTML shall be either the UTF-8 or UTF-16 transformation format of the Universal Multiple-Octet Coded Character Set (UCS) ISO 10646-1 [18]. The set of characters that shall be displayable is defined in annex E, "(normative): Character set" on page 434.

### 8.5.1 Conformance considerations

#### 8.5.1.1 Document conformance

An XML document is a DVB-HTML conformant document if it conforms to all the general rules in section 8.5.1.1.1, "General rules." on page 74 and is either:

- a valid document against the DVB-HTML DTD.
- an invalid document made conformant by the clauses in 8.5.1.1.2, "Invalid but conformant documents" on page 75.

##### 8.5.1.1.1 General rules.

A DVB-HTML conformant document shall be well-formed.

A DVB HTML conformant document should include an XML declaration.

A DVB-HTML conformant document shall respect the following clauses:

- a) The root element shall be <html>.
- b) The root element of the document shall designate the XHTML namespace using the xmlns attribute defined in Namespaces in XML [89]. The namespace designator for XHTML is "http://www.w3.org/1999/xhtml".
- c) There shall be a DOCTYPE declaration in the document prior to the root element. The public identifier included in the DOCTYPE declaration shall be present and should reference the DTD found in annex AA "(normative): DVB-HTML 1.0 DTD" using its Formal Public Identifier (FPI).

The FPIs that reference the DVB-HTML DTD shall be:

```
"-//DVB//DTD XHTML DVB-HTML x.y//EN"
```

Where values of x and y are found in Table 7, "Version identifiers" on page 75 for this version of the DVB-HTML 1.0 DTD

The following system identifiers are guaranteed to locate the above DTD. Other system identifiers may be used to locate the DVB DTD if required. Implementations of this specification are required to work correctly without fetching any DTDs from this location as part of their normal operation.

```
http://www.dvb.org/mhp/dtd/dvbhtml-x-y.dtd
```

Where values of x and y are found in Table 7, "Version identifiers" on page 75 for this version of the DVB-HTML 1.0 DTD

**Table 7 : Version identifiers**

	<b>x</b>	<b>y</b>
<b>value</b>	1	0

The following DOCTYPE declaration example indicates recommended practice in DVB-HTML documents:

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//DVB//DTD XHTML DVB-HTML 1.0//EN"
"http://www.dvb.org/mhp/dtd/dvbhtml-1-0.dtd"
>
```

#### 8.5.1.1.2 Invalid but conformant documents

The MHP platform is designed to support future compatibility and the possible presence of proprietary extensions; this specification defines how this is done in a way that authors can be sure of the behaviour of an MHP implementation that does not support the extension. Therefore, in addition to the support of the general rules, a DVB-HTML conformant document is also defined by one or more of the following conditions:

- An invalid document with respect to the DTD found in annex AA "(normative): DVB-HTML 1.0 DTD" and for which invalidity is only due to the presence of additional elements and attributes in a different namespace to the XHTML or DVB-HTML namespaces, and which does not define a new default namespace (see [Namespaces in XML \[95\]](#)).
- An invalid document with respect to the DTD found in annex AA "(normative): DVB-HTML 1.0 DTD" with an FPI in the document type declaration identifying a strictly later version of DVB-HTML than that defined in this specification and with additional elements or attributes in the XHTML or DVB-HTML namespaces, which, when removed, results in a document that would validate against the DTD found in annex AA "(normative): DVB-HTML 1.0 DTD".

A non-conformant document is one which is not valid with respect to the DTD found in annex AA "(normative): DVB-HTML 1.0 DTD", and is not made conformant by the above clauses.

Handling of non-conformant DVB-HTML documents is specified in section 8.5.1.2.1, "Error handling" on page 77

A document that is DVB-HTML conformant but not valid with respect to the DTD found in annex AA "(normative): DVB-HTML 1.0 DTD", is handled as detailed in section 8.5.1.2.2, "Handling of invalid but conformant documents" on page 77.

#### 8.5.1.2 DVB-HTML user agent conformance

The DVB-HTML user agent shall be conformant with the XHTML Family User Agent conformance clauses defined in [Modularization of XHTML \[80\]](#).

Additionally, before attempting to use it as part of an application the DVB-HTML user agent shall assess the conformance of:

- any received XML document signalled in the AIT as being part of the DVB-HTML application.
- any XML document whose document type declaration contains one of the DVB-HTML FPIs (see 8.5.1.1, "Document conformance" on page 74).

For the purpose of validation, it shall:

- remove from the XML document any elements and attributes in a different namespace than the XHTML or DVB-HTML namespaces.
- if the FPI of the document indicates that the version of the DVB-HTML document strictly superior to the version of the DVB-HTML language the implementation supports: remove any elements and attributes in the DVB-HTML or default XHTML namespace not present in the supported DVB-HTML DTD.

It shall subsequently validate (see clause 5.1 in XML 1.0 [65]) the resulting XML document against the DTD found in annex AA "(normative): DVB-HTML 1.0 DTD". Validation is performed by checking that the document does not violate any validity constraints as listed in table 8 "List of applicable validity constraints". Any violation report indicates the non conformance of the received XML document. Corresponding error handling is described in section 8.5.1.2.1, "Error handling" on page 77.

The way this process is performed is implementation dependent.

The document conformance shall be assessed before the generation of the DVBDOMStable event (see 8.11.2.3.2 on page 132). This conformance status shall be re-assessed by the DVB-HTML user agent in the presence of modification of the DOM tree. The way this constraint is checked is implementation dependent, but the invariant must be that the document continues to be conformant. If a document becomes at any point non-conformant, it shall be handled as described in section 8.5.1.2.1.2, "On DOM mutation" on page 77.

The DVB-HTML user agent is not required to be a full validating processor (according to W3C terminology in XML 1.0 [65]), in the sense that it is not required to be able to read and process any arbitrary DTDs and external parsed entities referenced in the document. It is also not required to handle an internal DTD subset attached to the document type declaration. For that reason, only the following XML validity constraints (see XML 1.0 [65]) are required to be checked:

**Table 8 : List of applicable validity constraints**

Validity constraint	Required
Root element type	Yes
Proper Declaration/PE Nesting	
Standalone Document Declaration	(note 1)
Element Valid	Yes
Attribute Value Type	Yes
Unique Element Type Declaration	
Proper Group/PE Nesting	
No Duplicate Types	
ID	Yes
One ID per Element Type	
ID Attribute Default	
IDREF	Yes
Entity Name	
Name Token	Yes
Notation Attributes	
One Notation Per Element Type	
No Notation on Empty Element	
Enumeration	Yes
Required Attribute	Yes
Attribute Default Legal	
Fixed Attribute Default	Yes
Proper Conditional Section/PE Nesting	
Entity Declared	Yes (note 2)
Notation Declared	
Unique Notation Name	
NOTE 1: the user agent is not required to fetch and interpret any external mark-up declarations.	
NOTE 2: only the requirement on entity reference matching is applicable. A DVB-HTML user agent is not required to check the XML requirements on parameter-entity or general entities declarations locations.	



### 8.5.1.2.1 Error handling

The user agent shall not present or allow interaction with a document which is not conformant (see 8.5.1.1, "Document conformance" on page 74). The "DVBDOMStable event" on page 132 shall not be generated for this document.

#### 8.5.1.2.1.1 On receipt of a document

If the non-conformant document is the root document of a DVB-HTML application, this application shall not be started, i.e. the corresponding DVB-HTML actor shall go from the Loading state to the Killed state. If the non-conformant document is not the root document, then the DVB-HTML application shall remain in the current state and the non-conformant document is treated as an unreachable resource.

During the parsing of a document, the DOM structure will necessarily be incomplete. No DOMExceptions indicating invalid structure will be thrown during this phase, and authors should make no assumptions on which elements will be available. See 8.11.2.3.2, "DVBDOMStable event" on page 132.

If an attempt is made to alter the DOM structure using the DOM APIs prior to delivery of the DVBDOMStable event then the DOMException with the error code INVALID\_STATE\_ERR shall be raised.

#### 8.5.1.2.1.2 On DOM mutation

A DVBException with the error code NON\_CONFORMANT\_ERR may be raised if an attempted modification of the DOM would result in a non-conformant document. The DOM shall remain unchanged on generating the exception (see 8.11.5, "DVB Exceptions" on page 152).

NOTE: The DOM allows for a DocumentFragment object for construction of partial intermediate structures which are not required to be valid or conformant.

Documents generated using `document.write` shall be handled as defined in 8.5.1.2.1.1, "On receipt of a document" on page 77, on closing the document.

### 8.5.1.2.2 Handling of invalid but conformant documents

When using only the default style sheet, the user agent shall not present or allow interaction with elements that make a conformant document invalid. Such elements are removed from presentation by the default style sheet (see annex AB, "(normative): DVB HTML StyleSheet" on page 1280), however the elements shall be retained in the DOM.

Implementations supporting such elements, or documents using them, may override the stylesheet to cause them to be presented to the user. The XHTML Family User Agent conformance clauses require that the text content of any unrecognised elements be presented to the user (unless a stylesheet changes this), so the stylesheet should not be overridden unless it is certain that the elements are supported and/or have no text content.

Attributes that make a conformant document invalid may be ignored by the user agent.

## 8.5.2 Set of modules required by this specification

The DVB-HTML document type definitions are made up of the following abstract modules indicated in table 9 below. The modules are defined in [p.15] of Modularization of XHTML [80]. An implementation of the DTD is defined in annex AA "(normative): DVB-HTML 1.0 DTD".

**Table 9 : Required modules for DVB-HTML (Sheet 1 of 2)**

Modularization	Required
Structure	Yes
Text	Yes
Hypertext	Yes
List	Yes
Applet	
Presentation	Yes
Edit	
Bi-directional Text	Yes

**Table 9 : Required modules for DVB-HTML (Sheet 2 of 2)**

Modularization	Required
Basic Forms	
Forms	Yes
Basic Tables	Yes
Tables	
Image	Yes
Client side Image map	Yes
Server side image map	
Object	Yes
Frames	Yes
Target	Yes
Iframe	Yes
Intrinsic events	
DVB Intrinsic events	Yes
Metainformation Module	Yes
Scripting	Yes
Style sheet	Yes
Style Attribute	Yes
Link	Yes
Base	Yes
Name Identification	
Legacy	

### 8.5.3 Semantics for modules

#### 8.5.3.1 XHTML modules

For the modules defined by [Modularization of XHTML \[80\]](#) the semantics are as defined there except as elaborated below and in 8.6, "Media Types" on page 83.

##### 8.5.3.1.1 Structure

Media types for which semantics have been specified when used in the profile link in the <head> element are defined in 8.6.2, "MIME media type use restrictions" on page 84.

##### 8.5.3.1.2 Text

Media types for which semantics have been specified when used in the `cite` attribute in the <q> element are defined in 8.6.2, "MIME media type use restrictions" on page 84.

##### 8.5.3.1.3 Hypertext

Media types for which semantics have been specified when used in the `href` attribute (hypertext links) in the <a> element are defined in 8.6.2, "MIME media type use restrictions" on page 84.

##### 8.5.3.1.4 Presentation

NOTE: The HTML conformance clause (see [HTML 4 \[90\]](#)) for this module does not require text to be rendered with a different appearance when these elements are used in the absence of other styling information.

### 8.5.3.1.5 Forms

Media types for which semantics have been specified when used in the `action` attribute in the `<form>` element and the `src` attribute in the `<input>` element are defined in 8.6.2, "MIME media type use restrictions" on page 84

NOTE: Authors should be aware that the MHP terminal may not have a user visible file system and therefore should not make any assumptions about availability of a file selection dialogue. This concerns the use of the `file` value of the `type` attribute of the `<input>` element.

### 8.5.3.1.6 Client-side Image Map

Semantics on the way to associate an image map with an element are defined in section 13.6.1 in HTML 4 [90] with the following exception:

- the URI in the `usemap` attribute shall point to the identifier of the `<map>` element as set by the `id` attribute.

If no `<map>` element is identified through the `usemap` attribute, then no association shall be assumed. This is in line with the module DTD implementation as defined in Modularization of XHTML [80]. In HTML 4 [90] this association was performed using the `name` attribute of the `<map>` element.

### 8.5.3.1.7 Image

Media types for which semantics have been specified when used in the `longdesc` and `src` attributes in the `<img>` element are defined in 8.6.2, "MIME media type use restrictions" on page 84

### 8.5.3.1.8 Object

Media types for which semantics have been specified when used in the `archive`, `classid`, `codebase` and `data` attributes in the `<object>` element are defined in 8.6.2, "MIME media type use restrictions" on page 84.

When the referenced object is an MPEG stream, see 8.6.7, "MPEG Audio" on page 87 and 8.6.8, "MPEG Video" on page 88.

When the referenced object is a DVB service, see 8.6.9, "DVB Services" on page 89.

When the referenced object is an Xlet see 8.9, "Xlet integration" on page 120.

For other media types, see 8.6.10, "Graphics content" on page 89.

### 8.5.3.1.9 Frames

Media types for which semantics have been specified when used in the `src` attributes in the `<frame>` element are defined in 8.6.2, "MIME media type use restrictions" on page 84.

### 8.5.3.1.10 Target

The semantics of specifying target frame information is described in section 16.3 in HTML 4 [90] with the following exception:

- the value of the `target` attribute shall point to the identifier of the frame as set through the `id` attribute.

If a target attribute refers to an unknown frame, then no action shall be performed.

NOTE: In HTML 4 [90], this was performed through the `name` attribute. Appendix B.8 in HTML 4 [90] is not relevant any more.

Use of target names are also subject to security considerations, see 8.14.3.1, "Restrictions on DOM elements introduced for security" on page 172.

### 8.5.3.1.11 Iframes

Media types for which semantics have been specified when used in the `src` and `longdesc` attributes in the `<iframe>` element are defined in 8.6.2, "MIME media type use restrictions" on page 84.

### 8.5.3.1.12 Metainformation

See 8.10.2.1, "ECMAScript APIs for accessing DVB-J" on page 122.

### 8.5.3.1.13 Scripting

Media types for which semantics have been specified when used in the `src` attribute in the `<script>` element are defined in 8.6.2, "MIME media type use restrictions" on page 84.

### 8.5.3.1.14 Link

Only the following `LinkTypes` are required to be supported in DVB-HTML for the `rev` and `rel` attributes:

**Alternate:** Designates substitute versions for the document in which the link occurs. When used together with the `lang` attribute, it implies a translated version of the document. When used together with the `media` attribute, it implies a version designed for a different medium (or media). Without one of these other attributes the value may be ignored.

**Stylesheet:** Refers to an external style sheet.

Style Sheets can also be associated with a DVB-HTML document by using a processing instruction whose target is `xml-stylesheet` as defined in "Associating Style Sheets with XML documents". This processing instruction follows the behaviour of the `<link rel="stylesheet">` semantics.

**Start:** Refers to the first document in a collection of documents. Use of this information is user agent specific.

**Next:** Refers to the next document in a linear sequence of documents. Use of this information is user agent specific.

**Previous:** Refers to the previous document in an ordered series of documents. the synonym "Prev" may also be used. Use of this information is user agent specific.

**Help:** Refers to a document offering help (more information, links to other source information, etc.). Navigation to this resource is user agent specific.

### 8.5.3.1.15 Base

Any media type is allowed for the `href` attribute in the `<base>` element since it is the structure of the `href` itself which is relevant, and not the resource referenced (which need not be accessed, or even present).

## 8.5.3.2 XHTML attributes

### 8.5.3.2.1 Longdesc, alt and cite attributes

DVB-HTML user agents shall make available a mechanism to access the alternate text or referenced resource and present it to the user. The specific means are implementation dependent.

### 8.5.3.2.2 Accesskey attribute

The `accesskey` attribute of HTML 4 [90] assigns an access key to an element. An access key is a single character from the document character set, which if pressed gives focus to the element, the semantics of giving focus to an element are element specific (see HTML 4 [90] section 17.11.2).

In order to have some measure of compatibility with the JAVA AWT JAE 1.1.8 API [31] and HAVi HAVi [50] event sets, the DTD used by the user agent points to an implementation specific DVB character entities module which shall define a set of XML entities which map all of the JAVA AWT JAE 1.1.8 API [31] and HAVi HAVi [50] "VK\_" events onto implementation specific characters.

The specific range of characters mapped to the "VK\_" entities is implementation specific, but shall not overlap with any character range that might be generated by an end user (e.g. on an optional attached keyboard).

So, for example given the following text in DVB-HTML:

```
<A accesskey="&VK_GUIDE;" href="guide/contents.html">
Guide information
</A>
```

The user agent shall, on receipt of the device specific operation that would generate a VK\_GUIDE event, give the focus to the <a> element (which in this case would cause the link to the guide content to be traversed).

It is not required for implementations to generate "VK\_" events other than the minimum set specified in table G.3, "Minimum set of input events" on page 446. But the user agent shall allow documents to refer to any of them. Generation of the "VK\_" events is allowed to be contextual, for example, when a text area element has the focus the "VK\_0" event may not be generatable by the user, since the device specific operation may generate a "0" character instead.

**NOTE:** The range of Unicode values 0xFB50..0xFDFF is unlikely ever to be used for legitimate input characters as they are presentations forms that don't belong in the Unicode coding space and could be used as a coding set.

The following is a partial example DTD fragment that defines the minimum set of entities:

```
<!ENTITY % VK_0 "&#01">
<!ENTITY % VK_1 "&#02">
<!ENTITY % VK_2 "&#03">
<!ENTITY % VK_3 "&#04">
<!ENTITY % VK_4 "&#05">
<!ENTITY % VK_5 "&#06">
<!ENTITY % VK_6 "&#07">
<!ENTITY % VK_7 "&#08">
<!ENTITY % VK_8 "&#09">
<!ENTITY % VK_9 "&#0A">
<!ENTITY % VK_UP "&#0B">
<!ENTITY % VK_DOWN "&#0C">
<!ENTITY % VK_LEFT "&#0D">
<!ENTITY % VK_RIGHT "&#0E">
<!ENTITY % VK_ENTER "&#0F">
<!ENTITY % VK_TELETEXT "&#10">
<!ENTITY % VK_COLORED_KEY_0 "&#11">
<!ENTITY % VK_COLORED_KEY_1 "&#12">
<!ENTITY % VK_COLORED_KEY_2 "&#13">
<!ENTITY % VK_COLORED_KEY_3 "&#14">
```

### 8.5.3.3 DVB-HTML modules

#### 8.5.3.3.1 DVB Intrinsic events

DVB Intrinsic events are attributes that are used in conjunction with elements that can have specific actions occur when certain events are performed by the user. The attributes indicated in the following table are added to the attribute set for their respective elements. Attributes defined by this module are:

**Table 10 : Elements of DVB Intrinsic events**

Elements	Attributes
body&	onload (Script), onunload (Script) (see 8.10, "Scripting" on page 122). onvbdomstable (Script) (see 8.11.2.3.2, "DVBDOMStable event" on page 132).
frame&	onload (Script), onunload (Script) (see 8.10, "Scripting" on page 122). onvbdomstable (Script) (see 8.11.2.3.2, "DVBDOMStable event" on page 132).

If these attributes are used then they must be within the scope of a namespace definition, namely "http://www.dvb.org/mhp". The namespace prefix shall be declared on the element of the document that uses them. And the namespace shall be "dvbhtml" (see [Namespaces in XML \[89\]](#)).

The following is an example of how these attributes should be used.

```
<?xml version="1.0"?>
<!DOCTYPE html
PUBLIC "-//DVB//DTD XHTML DVB HTML 1.0//EN"
"http://www.dvb.org/mhp/dvb/dvbhtml-1-0.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>Namespace vs. Intrinsic Events Example</title>
  <script type="text/ecmascript" src="setupEventHandlers.js" />
</head>
<body xmlns:dvbhtml="http://www.dvb.org/mhp" dvbhtml:ondvbdomstable="setupEventListeners()">
  <!-- rest of document here -->
</body>
```

## 8.6 Media Types

A DVB-HTML user agent shall be able to recognize and handle (in accordance with the appropriate conformance clauses see 7, "Content formats" on page 64) the following media types:

**Table 11 : Content types accessible in DVB-HTML**

MIME media type	Common name
text/xml	XML
application/xml	
text/css	CSS
text/plain	Monomedia format for text
text/dvb.utf8	
audio/mpeg	Monomedia format for audio clips or Audio
image/jpeg	JPEG
image/png	PNG
image/gif	GIF
image/mpeg	MPEG-2 I-Frames
video/dvb.mpeg.drip	MPEG-2 Video drips
video/mpeg	MPEG video
multipart/dvb.service	Multipart DVB Service
application/dvb.pfr	Downloadable Fonts
application/dvbj	Initial class file of an Xlet
text/ecmascript	ECMAScript

Resources of these MIME media types can be referenced using the locators defined in 14, "System integration aspects" on page 353 and 8.16.5, "DVB-HTML specific locators" on page 181 subject to the usage conditions below.

NOTE: User agents are recommended to be prepared to handle the eventuality that the MIME media type is incorrectly indicating the media type, alternative strategies (e.g. based on the filename, and inspecting the contents) should be employed. In addition a resource labelled as a specific MIME media type may or may not be conformant with the subsetting restrictions imposed by this specification (see 15.1, "PNG - restrictions" on page 363, 15.3, "JPEG - restrictions" on page 364). How the User Agent detects and handles this is implementation specific.

### 8.6.1 Uses of MIME media types

**%ContentType.datatype;** The DVB HTML DTDs define the entity "%ContentType.datatype;", for use in attribute declarations where MIME media types may be used. The following table lists these attributes along with the XHTML Modules in which they occur.

**Table 12 : Use of ContentType attribute on elements**

elem.attr	XHTML Modules
a.type	Hypertext
link.type	Link
object.type	Object
object.codetype	Object
param.type	Object
form.enctype	Basic Forms, Forms
style.type	Stylesheet
script.type	Scripting

**%URI.datatype**; The DTDs also define "%URI.datatype;", describing where URIs shall be supported. The following table lists those attributes with the relevant XHTML Modules for reference.

**Table 13 : Use of URI attribute on elements**

elem.attr	XHTML Module(s)
a.href	Hypertext
area.href	Client-side Image Map
base.href	Base
blockquote.cite	Basic Text
form.action	Basic Forms, Forms
frame.longdesc	Frames
frame.src	Frames
head.profile	Structure
iframe.longdesc	Iframe
iframe.src	Iframe
img.src	Image
img.longdesc	Image
img.usemap	Client-side Image Map
input.src	Basic Forms, Forms
input.usemap	Basic Forms, Forms
link.href	Link
object.archive	Object
object.classid	Object
object.codebase	Object
object.data	Object
object.usemap	Client-side Image Map
q.cite	Basic Text
script.src	Scripting

## 8.6.2 MIME media type use restrictions

The behaviour of the MHP terminal is only specified where there is an "X" at the intersection of element.attribute and MIME media type in table 14 "Use of MIME media type by element".

**Table 14 : Use of MIME media type by element (Sheet 1 of 2)**

Element. attribute	MIME media type														
	text/xml	application/xml	text/css	text/plain	text/dvb-utf8	audio/mpeg	image/jpeg	image/png	image/gif	image/mpeg	video/mpeg	multipart/dvb.service	application/dvb	video/dvb.mpeg.drip	text/ecmascript
As root element	X														
a.type	X					X						X	X		
link.type (note 3)	X		X												
object.type	X					X	X	X	X	X	X	X		X	
object.codetype													X		
param.type	(note 3)														



Table 14 : Use of MIME media type by element (Sheet 2 of 2)

Element. attribute	MIME media type														
	text/xml	application/xml	text/css	text/plain	text/dvb.utf8	audio/mpeg	image/jpeg	image/png	image/gif	image/mpeg	video/mpeg	multipart/dvb.service	application/dvb	video/dvb.mpeg.drip	text/ecmascript
style.type			X												
script.type															X

NOTE 3: Informative, authoring guideline: MIME media type constraints only apply to this attribute when the param.valuetype attribute has value "ref". No allowed object types required to be supported by this specification make use of this "type" attribute.

The following table shows whether the resource identified by the URI value of each element.attribute pair (vertical axis) has DVB defined semantics for each MIME media type (horizontal axis) in DVB-HTML; "X" = Yes, blank = No.

Table 15 : MIME media type usage (Sheet 1 of 2)

Element. attribute	exit:	MIME media type													
		text/xml	application/xml	text/css	text/plain	text/dvb.utf8	audio/mpeg	image/jpeg	image/png	image/gif	image/mpeg	video/mpeg	multipart/dvb.service	application/dvb	video/dvb.mpeg.drip
a.href	X <sup>a</sup>	X <sup>d</sup>				X <sup>b</sup>						X <sup>e</sup>	X <sup>d</sup>		
area.href	X <sup>a</sup>	X <sup>d</sup>				X <sup>b</sup>						X <sup>e</sup>	X <sup>d</sup>		
base.href		X (note 1)													
blockquote.cite					X <sup>f</sup>										
form.action (see 8.6.13)															
frame.longdesc					X <sup>f</sup>										
frame.src		X <sup>g</sup>													
head.profile															
iframe.longdesc					X <sup>f</sup>										
iframe.src		X <sup>g</sup>													
img.src							X <sup>h</sup>	X <sup>h</sup>	X <sup>h</sup>	X <sup>h</sup>	X <sup>c</sup>			X	
img.longdesc					X <sup>f</sup>										
img.usemap		X <sup>d</sup>													
input.src							X <sup>h</sup>	X <sup>h</sup>	X <sup>h</sup>	X <sup>h</sup>	X <sup>c</sup>			X	
input.usemap		X <sup>d</sup>													
link.href		X <sup>d</sup>		X <sup>j</sup>											
object.archive		(note 2)													
object.classid													X <sup>d</sup>		

Table 15 : MIME media type usage (Sheet 2 of 2)

Element. attribute	exit:	MIME media type														
		text/xml	application/xml	text/css	text/plain	text/dvb.utf8	audio/mpeg	image/jpeg	image/png	image/gif	image/mpeg	video/mpeg	multipart/dvb.service	application/dvbj	video/dvb.mpeg.drip	text/ecmascript
object.codebase		(note 1)														
object.data		X <sup>d</sup>				X <sup>b</sup>	X <sup>h</sup>	X <sup>h</sup>	X <sup>h</sup>	X <sup>h</sup>	X <sup>c</sup>	X <sup>e</sup>			X <sup>h</sup>	
object.usemap		X <sup>d</sup>														
q.cite					X <sup>f</sup>											
script.src																X <sup>i</sup>
NOTE 1: MIME type is irrelevant here as it's just the URI that matters, not the thing it identifies.																
NOTE 2: Specific to the object.																

### 8.6.3 Semantics of media type

- a) See 8.16.5.2, "Exit locator" on page 182.
- b) See 8.6.7, "MPEG Audio" on page 87
- c) See 8.6.8, "MPEG Video" on page 88
- d) See 8.6.5, "Application content" on page 86.
- e) See 8.6.9, "DVB Services" on page 89.
- f) See 8.5.3.2.1, "Longdesc, alt and cite attributes" on page 80
- g) See 8.6.4, "Frame content" on page 86
- h) See 8.6.10, "Graphics content" on page 89
- i) See 8.6.11, "Script content" on page 89
- j) See 8.6.12, "Style sheet content" on page 89

### 8.6.4 Frame content

The contents of a frame are restricted to be conformant DVB-HTML documents.

NOTE An implementation is not required to provide a scrolling mechanism for frames.

### 8.6.5 Application content

#### 8.6.5.1 When referenced via an AIT locator

An AIT locator is a locator of the form "dvb://current.ait/..." which is used to refer to application content via the AIT signalling. Using this form of locator, DVB-HTML applications are able to refer to currently available applications (see 11.7.2, "Application discovery and launching APIs" on page 261). For example:

```
<a type="application/dvbj" href="dvb://current.ait/Orgid.Appid?arg_0=val">
  Click here to launch application
</a>
```

The literal text of the locator, and not the translated form is stored in the DOM.

Activating such a link shall request the application manager to start the application referenced by this URL, passing to that application the specified parameters. The type attribute is used as a hint in the case that there are multiple application types with the same AppID and OrgID and should give preference to applications of the indicated type. The application shall be started depending on machine resources etc.

NOTE: Starting the new application may involve terminating the launching application.

This activation will fail as if the resource were not found if the launching application does not have the authority to start applications.

### 8.6.5.2 When not referenced via an AIT locator

Other references to DVB-HTML documents or fragments content can occur as a result of:

**href:** The semantics are as defined in [HTML 4 \[90\]](#).

**usemap:** The referenced element defines a map to overlay on the image. The semantics are as defined in [HTML 4 \[90\]](#).

**object.data:** The referenced document is rendered within the object as if it were a frame. The semantics are as defined in [HTML 4 \[90\]](#).

**src:** The referenced document is rendered within a frame or iframe. The semantics are as defined in [HTML 4 \[90\]](#).

## 8.6.6 Relative linking

The MIME media types allowed for this attribute depend on the value of the `link.rel` attribute.

For `link.rel="stylesheet"`, only the type `"text/css"` is defined and the user agent shall interpret this as specified in "14.3.2 Specifying external style sheets" of [HTML 4 \[90\]](#).

For all other values of `link.rel`, the resource pointed to shall be a conformant DVB-HTML document. The user agent may use this information in implementation specific ways (e.g. to prefetch the referenced document)

## 8.6.7 MPEG Audio

DVB HTML shall be able to reference MPEG audio as indicated in table "Use of MIME media type by element" on page 84. The referenced audio may be of either indefinite (see 7.2.1, "Audio" on page 66) or definite duration (see 7.1.4, "Monomedia format for audio clips" on page 66).

### 8.6.7.1 Resources of indefinite duration

Links to an MPEG elementary stream of indefinite length carried in the broadcast as specified in 7.2.1, "Audio" on page 66 shall select the service component for presentation into the current service context. If an audio service component is currently playing, it is replaced in its entirety by the specified selection

If the reference requires specific interaction to invoke (i.e. `a.href`, `area.href`) then the audio selection shall be after the reference is actioned. An implementation may impose a short delay before audio is audible, but this shall not be so long as to make the new audio appear unconnected to the user action.

If the reference does not require a specific reference to invoke (i.e. `object.archive`, `object.data`) on an object with the content type of `"audio/mpeg"`, then the audio stream shall be selected at an implementation specific time after the containing document becomes active. Any `<param>` elements included in such objects shall be ignored.

#### 8.6.7.1.1 Relation to document events

In the case of non-actioned references, the following constraints shall be observed:

- The `DVBDOMStable` event shall be generated prior to playing the stream.
- The media should be rendered as soon as possible after the `DVBDOMStable` event.
- The `load` event shall only be generated after the stream is playing

### 8.6.7.2 Resources of definite duration

Referencing an audio file resource of finite length (e.g. carried in DSMCC carousel), that carries audio data as specified in 7.1.4, "Monomedia format for audio clips" on page 66: shall cause audio to be rendered according to the restrictions in G.2, "Audio" on page 444.

If the reference requires specific interaction to invoke (i.e. `a.href`, `area.href`) then the audio shall be rendered once in its entirety after the reference is actioned. An implementation may impose a short delay before audio is audible, but this shall not be so long as to make the audio appear unconnected to the user action.

If the reference does not require a specific reference to invoke (i.e. `object.archive`, `object.data`) on an object of type "audio/mpeg", then by default the audio is rendered once at an implementation specific time after the containing document becomes active. The following `<param>` name, value pairs shall be implemented for audio of finite duration rendered in an object:

name	legal values	meaning
loop	(true   false)	When the audio data has been played in its entirety, then it should be played again from the beginning of its associated data, so as to cause a "seamless" continuous (infinite) audio playback until the document is unloaded.

#### 8.6.7.2.1 Relation to document events

In the case of non-actioned references, the following constraints shall be observed:

- The `DVBDOMStable` event and the `load` event shall be generated prior to playing the audio.

## 8.6.8 MPEG Video

NOTE: MPEG video or images may only be displayable full- or quarter-screen: if authors ask an object to display it in a way the hardware doesn't support, the user agent should follow the usual fallback rules for unavailable resources in object. Authors may need to provide alternate content to cover this, probably also fixing the size to prevent layout differences. However, the `img` and `input` tags do not provide this fallback capability, so authors should be more careful in using them, to be sure that all target STBs will display the image as desired.

### 8.6.8.1 Video Resources of indefinite duration

Referencing an MPEG elementary stream of indefinite length carried in the broadcast as specified in 7.2.2, "Video" on page 66 shall cause the referenced video to be placed in the bounding rectangle of the element after styling is applied (see 8.8.4.2.1.3, "clip-video" on page 105). Access to the referenced video stream shall not cause tuning to occur, if this would cause the implementation to lose access to the service carrying the application.

DVB HTML defines only the case where the reference does not require user interaction to invoke (i.e. `img.src`, `input.src`, `object.archive`, `object.data`), if access to the stream requires tuning, the referenced video stream is tuned to at an implementation specific time after the containing document becomes active.

Video referenced by an `<object>`, `<img>` or `<input>` is considered to be referenced later than that in external style sheets, and thus takes precedence over the selected background video in the case that there are insufficient resources to display both the referenced video and the background video simultaneously (in accordance with 11.2.9, "Intra application media resource management" on page 242). In the case of multiple references within a document, it is implementation specific which is considered referenced later.

No `<param>` elements are defined in for objects of type "mpeg/video".

### 8.6.8.1.1 Relation to document events

In the case of non-actioned references to video of indefinite length, the following constraints shall be observed:

- The `DVBDOMStable` event shall be generated prior to playing the video.
- The media should be rendered as soon as possible after the `DVBDOMStable` event.
- The `load` event shall only be generated after the stream is playing

#### 8.6.8.1.1.1 Modification of the referencing attribute using the DOM

Alterations to the video source due to script (e.g changing styling or `src` attributes) are considered as later references than any declarative reference. The playing video is replaced with the newly referenced video source using the same rules as for the original value of the attribute.

### 8.6.8.2 Resources of definite duration

This version of the specification does not define referencing of video segments of finite duration.

## 8.6.9 DVB Services

Use of URLs with that reference a service in contexts that do not require user action (i.e `object.data`, `object.archive`) on objects of type "multipart/dvb.service" are defined only for the media aspects of a service (e.g. video, audio, subtitles) and shall not cause service selection, but the media components of that service shall be displayed according to ("MPEG Audio" on page 87 and "MPEG Video" on page 88).

Links to DVB services in contexts that require user action (i.e. `a.href`, `area.href`) shall cause service selection to the referenced service if the application is authorised (see 8.15.1.9, "javax.tv.service.selection.

[ServiceContextPermission](#)" on page 174). If authorisation fails, then the link shall be treated as if referencing an unavailable resource.

Links to DVB services of the form "dvb:" referencing broadcast services, and other locators referencing stored services (see 9.7.2, "Stored services" on page 204) or services over the interaction channel shall be allowed (see 9.7.1, "Applications loaded from the interaction channel" on page 204).

NOTE: Selecting the new service can cause the selecting application to be terminated.

### 8.6.10 Graphics content

Graphics formats shall be supported as specified in 7.1, "Static formats" on page 64. Such graphics shall be displayed inline with the content in the box generated as a result of processing any CSS rules.

NOTE: Images of other types than the minimum requirements for the MHP may be referenced, but the implementation may not display them. Since the `img` and `input` tags do not provide fallback capability, authors should be careful in using them, to be sure that all target STBs will display the content as desired.

### 8.6.11 Script content

The referenced resource shall be in a plain text format admitted by this specification 7.1.5, "Monomedia format for text" on page 66. The content shall be ECMAScript source code as defined in [ECMAScript \[81\]](#), and shall be handled as if it were inline.

### 8.6.12 Style sheet content

The referenced resource shall be in a plain text format admitted by this specification 7.1.5, "Monomedia format for text" on page 66. The content shall be CSS source code as defined in [CSS 2 \[87\]](#), and shall be handled as defined by the cascading rules therein.

## 8.6.13 HTTP(S) URLs

For `form.action` only HTTP(S) URLs are valid (as form submission relies on GET/POST). The the user agent is only required to handle the case where the content returned is a conformant top-level document.

## 8.6.14 CSS Properties

### 8.6.14.1 Sources of MIME media type use points

**<uri>**: The CSS 2 specification defines the value type `<uri>`. For the places where `<uri>` is a permitted value in CSS see [CSS 2 \[87\]](#).

### 8.6.14.2 MIME media type use restrictions

Behaviour when referencing the following MIME media types is not specified for any of the `<uri>` type in CSS (but may nevertheless be the MIME media types of some parts of a DVB-HTML application):

- `text/xml`
- `application/xml`
- `text/css`
- `audio/mpeg`
- `text/dvb.subtitle`
- `image/dvb.subtitle`
- `application/dvbj`
- `text/ecmascript`

The following table shows whether the resource identified by the URI value of each CSS attribute (vertical axis) has DVB defined semantics for each MIME media type (horizontal axis) in DVB-HTML; "X" = Yes, blank = No.

**Table 16 : Usage of <uri> type in CSS**

CSS Attribute		MIME media Type									
		<code>text/plain</code>	<code>text/dvb.utf8</code>	<code>image/jpeg</code>	<code>image/png</code>	<code>image/gif</code>	<code>image/mpeg</code>	<code>video/mpeg</code>	<code>video/dvb.mpeg.drip</code>	<code>multipart/dvb.service</code>	<code>application/dvb.pfr</code>
<code>background-image</code>				x <sup>b</sup>	x <sup>b</sup>	x <sup>b</sup>	x <sup>b</sup>	x <sup>c</sup>	x <sup>c</sup>		
<code>content (note )</code>		x <sup>a</sup>									
<code>cursor</code>				x <sup>b</sup>	x <sup>b</sup>	x <sup>b</sup>	x <sup>b</sup>				
<code>list-style-image</code>				x <sup>b</sup>	x <sup>b</sup>	x <sup>b</sup>	x <sup>b</sup>				
<code>@font-face</code>	<code>src</code>										x
<code>@viewport</code>	<code>"background:"</code>			x <sup>b</sup>	x <sup>b</sup>	x <sup>b</sup>	x <sup>b</sup>		x <sup>c</sup>		
	<code>"background-video-n"</code>							x <sup>c</sup>		x <sup>d</sup>	
NOTE 1:											

a) See "Generated Content" on page 91

- b) See "Graphics styling" on page 91
- c) See "Video Styling" on page 91
- d) See "DVB Service styling" on page 91

## 8.6.15 Generated Content

The referenced resource shall be in a plain text format admitted by this specification 7.1.5, "Monomedia format for text" on page 66. The content shall be handled as if it were inline as defined in CSS 2 [87].

- NOTE: Informative, authoring guideline, in CSS2 you cannot:
- generate markup (only XML CDATA)
  - apply further style to generated content
  - access generated content via DOM

## 8.6.16 Graphics styling

Graphics formats shall be supported as specified in 7.1.1, "Bitmap image formats" on page 64. Such graphics shall be handled as defined in CSS 2 [87].

## 8.6.17 Video Styling

Video and video drips may be used as the `background-image` style property. The resource shall be handled as an image type as defined in CSS 2 [87]. Video is displayed as defined in 8.6.8.1, "Video Resources of indefinite duration" on page 88 as a reference that does not require user interaction.

- NOTE: MPEG video or images may only be displayable full- or quarter-screen: if you ask an object to display it in a way the hardware doesn't support, the user agent should follow the usual fallback rules for unavailable resources in object.

Video may be used as the viewport "`background-video-n`" style property. The resource shall be used to select the referenced video stream into service context, and cause it to be displayed in the corresponding background video plane.

Video drips may be used as the viewport "`background:`" property. The resource shall be used to fill the corresponding background plane.

## 8.6.18 DVB Service styling

Use of URLs with that reference a service as background-video styling are defined only for the media aspects of a service (e.g. video, audio, subtitles) and shall not cause service selection, but the only the media components of that service shall be displayed see 8.8.5.3.2, "The `@viewport` rule" on page 107.

## 8.7 Synchronization

### 8.7.1 Triggers Overview

Triggers provide a means by which the application provider can interact with an application running on an end user terminal. Although in principal this interaction could be delivered by any communication channel available to the MHP, this specification defines only a binding to DSM-CC stream events. Triggers are received by the user agent and used to raise events in a running DVB-HTML application.

Triggers are small messages sent in a broadcast separately from the main content, which may cause a change in the behaviour of applications that choose to register for them. Triggers can carry a time at which they should be delivered, plus a small amount of payload data which an application can register for.

Application authors can think in terms of traditional media time bases (e.g. SMPTE time codes) that are a simple frames/seconds offset from the beginning of the media. Triggers can signal that a point on the time base has been reached. The author has to "name" these events so that applications can subscribe to them. This name may be meaningful to the author as is illustrated in table 17:

**Table 17 :**

Event name	Event time
Start	00:00:00.00
End of introduction	00:00:30.00
End of recipe	00:05:00.00
End of recipe	00:08:00.00
Start or roll out	00:11:00.00
End	00:11:30.00

The behaviour following the event is implemented in the code of the application. It is possible that data is delivered with the event to modify the behaviour.

The following issues are described in more detail below:

- How triggers are transported (see 8.7.1.1 "Transport of triggers")
- How an application registers for and receives triggers (see 8.7.1.2 "Application registration and reception")

#### 8.7.1.1 Transport of triggers

We consider here transport of triggers in:

- DSM-CC stream events.

#### 8.7.1.2 Application registration and reception

In order to integrate with W3C event models, triggers are delivered to DVB-HTML applications as DOM events. Registration is through:

- Explicit registration using the DOM API

Delivery is by DOM event. Trigger DOM event interfaces inherit from the interface `Event` in DOM (see [Document Object Model \(DOM\) Level 2 Events Specification \[84\]](#)).

#### 8.7.1.3 Binding to DSM-CC Stream events

A mechanism is provided in section 8.7.3, "[Binding the event factory file to the application](#)" on page 99 that identifies the location of the DSMCC Stream Event message associated with any document entity of the DVB-HTML application that registers to a trigger event. However, it is possible to bypass this mechanism through a default location binding.



By default, a DVB-HTML application is bound to all DSMCC Stream Event messages which are located in the application root directory defined in the `physical_root` field in the `dvb_html_application_location` descriptor of the AIT. In that case, those DSMCC Stream Event messages list the events used in the context of the whole application. See section on default trigger mechanism (See 8.7.4, "Default Trigger Mechanism" on page 101).

## 8.7.2 Trigger Events

### 8.7.2.1 Converting stream events into DOM events

DVB-HTML triggering contains a mechanism by which the stream events are converted into DOM events. This has several features.

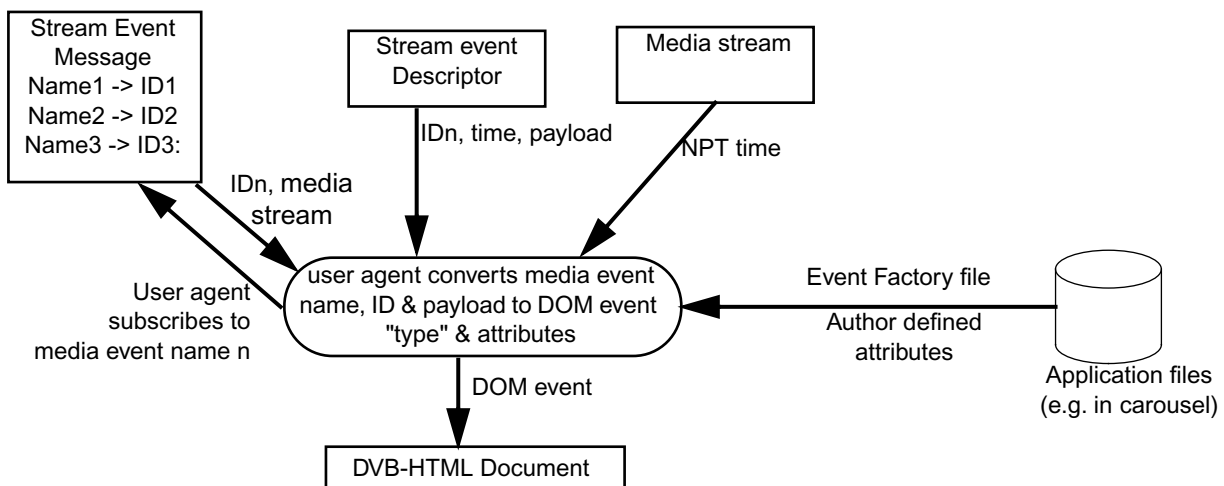
- A mechanism to override the default association between application and stream event message which provides a degree of abstraction between the transport mechanism of the stimulus aspect of the event and the author's behavioural view
- A mechanism which expands the media event into one or more DOM events, this includes mapping the media event name into a DOM type, and extracting any or all of the media event payload and converting it into properties of the DOM event

The application document receives the DOM events and has its behaviour modified by them using the standard mechanism of DOM event delivery (see 8.4 "Document Object Model (DOM) Level 2 Events Specification").

In order to describe the above, the application has associated with it an "event factory file" (See 8.7.2.2, "Event Factory File definition" on page 94). This file provides the information the user agent needs to locate stream event messages (if not the default) for the event name bindings, and determines which media event names to subscribe to. If not present, the default event factory file is assumed (See 8.7.2.4, "Default Event Factory File" on page 97). It can also determine which parts of the media event payload become properties of the resulting DOM event. Factory files are associated with resources of the DVB-HTML application using a "linkage file" (See 8.7.3, "Binding the event factory file to the application" on page 99)

On receipt of the stimulus associated with an event ID (e.g. the correct NPT time arrives) the user agent is able to "fire" the DOM event into the DVB-HTML application to be handled.

The entire process is summarised by the following data flow diagram:



**Figure 11 : Overview of the authored event mechanism**

The following text provides an informative description of the different steps involved when a DVB-HTML document subscribes to a trigger event delivered by DSM-CC stream events using a linkage file and factory file, an implementation may choose to perform some steps differently, however the overall result shall be equivalent:

On starting the application the user agent locates the linkage file. If no linkage file exists, then the default processing shall apply to this application (see 8.7.4, "Default Trigger Mechanism" on page 101).

On loading a document, the user agent locates any event factory files that apply to the document. If no factory files apply to this page, the default event factory file shall be applied (see 8.7.2.4, "Default Event Factory File" on page 97), the user agent collates all of the `trigger-event` elements in the event factory files.

If the DVB-HTML document, through an embedded Xlet or any attached script, implements an `EventListener` interface and subscribes to a trigger event of a type mentioned in any `trigger-event` element in the collated set, through a call to the DOM `addEventListener()` method on the element node of the document DOM. The user agent locates the DSMCC Stream Event message by following the URI specified by the `stream` attribute in the event element. The DSMCC Stream Event message provides the mapping between the event name in the `trigger-event` element and the event id and references the source of the stream event descriptors that the user agent needs to monitor. If the collated `trigger-event` elements do not mention a matching type then the default element shall be assumed (see 8.7.4, "Default Trigger Mechanism" on page 101).

On receipt of the stream event descriptor that provides the mapping between the event id and both the `eventNPT` and the associated payload, the user agent begins the synthesis of the DOM event, as defined by the event factory element. When the NPT of the referenced stream is equal to the `eventNPT` of the stream event descriptor plus any offset, or as soon as the stream event descriptor is received in case of 'do it now' events (see B.2.4.3, "DSM-CC Sections carrying Stream Descriptors" on page 407), the DOM event is injected in the DOM implementation at the node whose id matches the `target` attribute of the `trigger-event` element.

Payload of the stream event is parsed by the expression and placed in the synthesised DOM event using appropriate attribute names.

If following the DOM event propagation rules this event reaches the handler placed by the Xlet or script, the handler is activated and receives the synthesised DOM event.

## 8.7.2.2 Event Factory File definition

### 8.7.2.2.1 Syntax

The event factory file is an XML file with syntax defined by the following DTD.

```
<!-- ..... -->
<!--          DVB HTML Event Factory File DTD -->
<!-- ..... -->
<!-- file: htmleventfactoryfile-1-0.dtd-->
<!-- ..... -->

<!--
The following formal public identifier shall be used to identify this file:

    "-//DVB//DTD DVB HTML Event Factory 1.0//EN"

The following URL may be used to reference this file:

    http://www.dvb.org/mhp/dtd/htmleventfactoryfile-1-0.dtd

-->

<!ELEMENT htmleventfactoryfile (trigger-event|system-event|time-event)*>

<!ELEMENT trigger-event (event-attribute)*>
<!ATTLIST trigger-event
    stream          CDATA          #IMPLIED
    event           CDATA          #REQUIRED
    type            CDATA          #IMPLIED
    target          CDATA          #IMPLIED
    time            CDATA          "+0"
    bubbles         %Boolean;     "true"
    cancelable     %Boolean;     "true"
>

<!ELEMENT system-event (event-attribute)*>
<!ATTLIST system-event
    stream          CDATA          #IMPLIED
    event           CDATA          #REQUIRED
```

```

    system-event-type          CDATA          #REQUIRED
>
<!ELEMENT time-event (event-attribute)*>
<!ATTLIST time-event
  type          CDATA          #IMPLIED
  target        CDATA          #IMPLIED
  time          CDATA          "+12"
  bubbles       %Boolean;     "true"
  cancelable    %Boolean;     "true"
>

<!ELEMENT event-attribute EMPTY>
<!ATTLIST event-attribute
  attribute-name CDATA          #REQUIRED
  attribute-pattern CDATA       #REQUIRED
>

```

### 8.7.2.2.2 Element semantics

The elements of the event factory file have the following semantics:

**htmleventfactoryfile:** This element serves as the root element for an event factory file.

**trigger-event:** This element specifies a relationship between:

- a media stream event
- the DOM event interface to which it maps
- the DOM event destination.

**system-event:** This element specifies a relationship between a media stream event and the system event to be generated. See 8.7.2.6, "System events" on page 98.

**time-event:** This element specifies a relationship between an absolute time and

- the DOM event interface to which it maps
- the DOM event destination.

**event-attribute:** This element specifies the mapping of media event payloads into attributes of the trigger or system event.

The **trigger-event** and **time-event** element can contain zero or more **event-attribute** elements. These describe the construction of a derived interface from the DOM `Event` interface, which adds attributes.

The **system-event** element can contain zero or more **event-attribute** elements. These provide parameters to the system event.

#### 8.7.2.2.2.1 Duplicate elements

Each **trigger-event**, **time-event** or **system-event** generates an event regardless of whether any are effectively duplicates.

#### 8.7.2.2.3 Attributes semantics

The attributes of the event factory file have the following semantics:

**stream:** Names the stream event message, this shall be a "dvb:" URI and must resolve to a DSM-CC StreamEvent message. It may be relative to the application files or provide a full reference to the file system that carries the StreamEvent message. So, regardless of what file system carries the DVB-HTML application the implementation can locate the source of the events. If missing the default location is assumed.

**event:** The name of the media event in the stream event message which provides the stimulus for this event. Note that multiple broadcast-event records may reference the same media event, possibly with time offsets. This provides the names that the user agent must subscribe to.

**type:** The type of event which will be exposed to the DOM on triggering this event. If missing, the type will be the same as event.

**system-event-type:** The type of the system event which will be received by the user agent.

**target:** If this attribute is provided it specifies the ID of the target node in the HTML page. If the attribute is not provided then the target is the root of the DOM tree, also referred to as the document node. Any listeners on those event types should then be positioned on the document root node to be able to receive the event, or be positioned on the document node.

Example:

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//DVB//DTD XHTML DVB HTML 1.0//EN"
  "http://www.dvb.org/mhp/dtd/dvbhtml-1-0.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:dvbhtml="http://www.dvb.org/mhp">
  <head>
    <script type="text/ecmascript">
      // event listener declaration
      function handleEvent(evt) { /* Handle the event */ }

      // the listener is positioned on the document root node,
      // i.e. the html node
      function setupEventListeners () {
        var htmlNode = document.documentElement;
        htmlNode.addEventListener("myTriggerEvent", handleEvent, true);
      }
    </script>
  </head>
  <body dvbhtml:onload="setupEventListeners()" >
  </body>
</html>
```

**offset:** An optional offset time (in ms) which can be added to the NPT time of the stimulus to delay the firing of the event, this time is used to calculate a new NPT time for the delivery of the event.

**time:** The value of this parameter is detailed in 8.16.2, "Clock values" on page 179. Both Absolute and Offset times are allowed. If an offset form is given (A full, partial or timecode value with an initial '+' or '-' sign) then the event time is measured as an offset with respect to the UTC of the midday of the date attribute. Otherwise the value must lie in the range 0:0:0.0 to 23:59:59.99 (a value within one clock resolution of midnight), and specifies a clock time (in 24 hour format) on the day of the date attribute.

For example:

```
02:30:03    = 2 hours, 30 minutes and 3 seconds past midnight
+30:00:10.25 = 30 hours, 10 seconds and 250 milliseconds after the current midday
```

The MHP shall support offset referencing of at least +/- 12 hours.

**date:** The value of this parameter is detailed in 8.16.1, "Date Values" on page 179. It gives the date which on which the midday reference for the time attribute occurs. If not specified the value will be the date on which the application began executing.

**bubbles:** This attribute sets the DOM event attribute "bubbles" as defined in 84 "Document Object Model (DOM) Level 2 Events Specification".

**cancelable:** This attribute sets the DOM event attribute "cancelable" as defined in 84 "Document Object Model (DOM) Level 2 Events Specification".

**attribute-name:** The name which will be used for the DOM event attribute.

**attribute-pattern:** A regular expression which describes how to parse the data from the trigger payload in order to initialise the DOM event attribute. The attribute-pattern is of the form:

```
/pattern/flags:replacement
```

where pattern follows the grammar in 15.10.1 of [81], and flags are at most one each of ("g", "i" or "m"). The replacement text is arbitrary text, but may contain '\$' values which are substituted by elements collected by the pattern following the rules in 15.5.4.10 of [81]

For example `" / (\d+) /g : $1$2$3 "` would be a legal attribute-data value. And applied to the trigger payload:

```
"box 12 in 34 by 567 for 248"
```

would deliver the data:

```
"1234567"
```

### 8.7.2.3 Default Event Factory Element

Media events that are not defined in any event factory files associated with a DVB-HTML document and whose name "event-name" does not start with "dvb." are treated as if associated with a default event factory element which is defined with the following piece of syntax:

```
<trigger-event event = "event-name"
  type = "event-name"
  time = "0"
  bubbles = "true"
  cancelable = "true">
  <event-attribute
    attribute-name = "payload"
    attribute-pattern = "/*:$1"/>
</trigger-event>
```

**NOTE:** The event type and the media event name are identical in this case. It is indeed assumed here that every trigger event corresponds to a stream event with name value equals to the type of this trigger event. Since, the stream attribute is not present, the default location of the DSMCC StreamEvent messages is assumed. Also, since the target attribute is not present in this default event factory file, the target is the document root element (i.e. the html node).

Any listeners on those event types should then be positioned on the document root node to be able to receive the event, or to be positioned on the document node (see target attribute definition in 8.7.2.2, "Event Factory File definition" on page 94).

If the media event name value "event-name" starts with "dvb." then the media event is considered as being a system event, the default location of the DSMCC Stream Event message is assumed and the semantics of 8.7.2.6, "System events" on page 98 apply.

### 8.7.2.4 Default Event Factory File

In the absence of an event factory file associated with a DVB-HTML document, a default one is assumed. The default event factory file associated with a DVB-HTML document contains the definition of all the trigger events the document registers with. The aggregation of the default event factory elements constitutes the default event factory file for a DVB-HTML document.

### 8.7.2.5 Worked example

An applications event factory file contains the following text:

```
<trigger-event
  stream="dvb://233a..1234/goals"
  event="goal"
  type="score"
  cancelable="false">
```

```
<event-attribute
  attribute-name="score"
  attribute-data="\[s(core)?:(\d+)-(\d+)/$1-$2" />
</trigger-event>
```

This defines that the stream event message for this event is in the carousel location "dvb://233a..1234/goals" within this file will be a loop of event names, one of which should be "goal" this is the event that the user agent must subscribe to. The generated DOM event will be of type "score" to suit the naming convention of the author, and will not be cancelable. The media event will carry some payload data, and this should be placed in a DOM attribute called "score" in the resulting DOM event.

This will result in DOM events being delivered to the DVB-HTML application that have the following interface:

```
interface score : trigger-event {
  readonly attribute DOMString score;
  void               initScoreEvent(
    in DOMString typeArg,
    in boolean canBubbleArg,
    in boolean cancelableArg,
    in DOMString scoreArg );
};
```

Assume a stream event descriptor with the following payload bytes is received:

```
<http://xxx.yyy.com/fun.html>[score:1-2]
```

Then the resultant DOM event would be constructed using the following initialisation

```
initScoreEvent(
  "score", true, false, "1-2"
)
```

The following DOM event properties will be filled in the base class as follows:

- DOMTimeStamp timeStamp - this will contain the ms count since the 1 Jan 1970.

### 8.7.2.6 System events

System events use the same transport and binding mechanisms as [trigger-events](#), but instead of being converted into DOM events and being delivered to the DVB-HTML application they are entirely handled by the user agent. The mapping process from media event names to system event names is identical to that for [trigger-events](#), that is the [system-event](#) element defines the [stream](#) (optionally) and [event](#) name to subscribe to. However, in this case the [type](#) attribute defines the kind of system event. The following values for [type](#) are defined:

- "dvb.start"
- "dvb.page"

Note: All event names that start with "dvb." are reserved for use by DVB to avoid name clashes with possible other name definers.

The attributes associated with system events are fixed, but use the same mechanism to pull the information from the payload bytes.

#### 8.7.2.6.1 dvb.start event

- The dvb.start event causes a DVB-HTML application in the ACTIVE state to receive the AppStarting (see [8.11.2.2, "Lifecycle events" on page 129](#)) event. No payload data is defined for the start event: any [event-attribute](#) elements associated with this shall be ignored.
- The dvb.start events are only defined in the context of the DVB-HTML application entry point document (identified by the application location descriptor): if the start event element is present in the event factory file of any other DVB-HTML document it shall be ignored.
- More than one dvb.start binding can be defined but only the first to fire shall cause an AppStarting event.

The resultant event time of the start event specifies the time at which the application entry point document is intended to be displayed to the user. Ideally the implementation should begin fetching data and formatting the new page off-screen, and then display it at the time specified. It is not possible in general to predict how long any page might take to render, so that no matter how long in advance of the desired display time the page message is delivered there is no fixed guarantee an MHP will have completed the rendering.

### 8.7.2.6.2 dvb.page event

The **event-attribute** element associated with the dvb.page **system-event** allows the broadcaster to direct the application at a specific page during execution. Referencing pages that are not reachable due to security or other restrictions (see 8.16.3, "Unrealisable locators" on page 180) shall cause the event to be ignored. After processing of the event completes the application shall have a single window containing the referenced document. Even if the referenced document is currently being displayed it shall still be reloaded and redisplayed.

In order for an application author to guarantee that page events operate globally in an application they will have to use the broadest possible regular expression ("\*") in the linkage file to ensure that the event factory file that specifies the dvb.page event is bound to all DVB-HTML documents.

The following values for **attribute-name** are defined:

**Table 18 :**

<b>attribute-name</b>	<b>number of occurrences</b>
actuate	zero or one
href	one
title	zero or one

**actuate:** The results of the evaluation of the **attribute-pattern** should be one of the following:

- "onLoad"
- "onRequest"

the semantics of other values are not defined and shall be ignored. If this attribute is not present it is treated as if "onRequest".

If the result is "onLoad" then the user agent loads the page (as described above) given in the href attribute (as described below).

If the result is "onRequest", the user is informed that the document is available and chooses whether to switch to it. If the user accepts the behaviour is as if "onLoad". If the user declines the offer then the user presentation remains unchanged.

**href:** The results of the evaluation of the **attribute-pattern** yield a URI. The URI which the user agent should load.

**title:** The results of the evaluation of the **attribute-pattern** yields text that is intended to describe the target to the user in the "onRequest" interaction. The text encoding shall be compatible with 7.1.5, "Monomedia format for text" on page 66.

## 8.7.3 Binding the event factory file to the application

A DVB-HTML application is usually constructed of multiple DVB-HTML documents, each such file can have a different trigger event set to which it subscribes. Each file can be associated with zero or more event factory files. If no event factory file is associated with a DVB-HTML document then the default event factory file is assumed (See 8.7.2.4, "Default Event Factory File" on page 97). If no event element is associated with an event type in all the event factory files associated with a DVB-HTML document, then the default event factory element is assumed for this event (See 8.7.2.3, "Default Event Factory Element" on page 97).

If more than one event factory file is associated with a document then the behaviour is as if all of the elements were contained in a single file. The order of the files is not significant.

In order to define the mapping between DVB-HTML files and event factory files, each application is associated with zero or one "event linkage" file, this event linkage file defines the association between specific event factory files and the documents to which they relate. If there is no event linkage file then the default trigger mechanism is assumed (See 8.7.4, "Default Trigger Mechanism" on page 101).

### 8.7.3.1 Syntax of event linkage file

This file provides a list of zero or more event factory attribute files for each document in the application. The event linkage file is an XML file using the following DTD.

```
<!-- ..... -->
<!--          DVB HTML Event Linkage File DTD -->
<!-- ..... -->
<!-- file: htmeventlinkagefile-1-0.dtd-->
<!-- ..... -->

<!--
The following formal public identifier shall be used to identify this file:

    "-//DVB//DTD DVB HTML Event Linkage 1.0//EN"

The following URL may be used to reference this file:

    http://www.dvb.org/mhp/dtd/htmeventlinkagefile-1-0.dtd

-->

<!ELEMENT htmeventlinkagefile (linkage)*>

<!ELEMENT linkage (location)+>
<!ATTLIST linkage
    boundary          CDATA          #REQUIRED
>

<!ELEMENT location EMPTY>
<!ATTLIST location
    URI              CDATA          #REQUIRED
    language         CDATA          #IMPLIED
>
```

### 8.7.3.2 Semantics of event linkage file

**htmeventlinkagefile:** This element serves as the root element for an event linkage file.

**linkage:** This element specifies a set of DVB-HTML documents with which the event factory files identified by the location element are to be associated.

**boundary:** This is a regular expression identifying the set of documents. The encoding is identical to that used in the application boundary descriptor (see 10.10.3, "DVB-HTML application boundary descriptor" on page 233).

**location:** This element specifies the location of an associated event factory file using the URI.

**URI:** The URI.

**language:** This optional attribute identifies the user preference language required before the event factory file is interpreted. If this attribute is empty then event factory file applies regardless of user preference. If the user preference language does not match then the referenced event factory file does not apply. This matching is implementation defined.

The encoding of this attribute shall be ISO 639.2 [19].

### 8.7.3.3 Example

```
<?xml version="1.0"?>
<!DOCTYPE linkage "-//DVB//DTD DVB HTML Event Linkage 1.0//EN"
"http://www.dvb.org/mhp/dtd/htmeventlinkagefile-1-0.dtd">
```



```

<linkage boundary = "*" >
  <location URI = "./events/app_events.evt" />
</linkage>
<linkage boundary = "foo.htm" >
  <location URI = "http://www.xx.com/app/ev1.evt" />
  <location URI = "http://www.xx.com/app/ev2.evt" />
</linkage>
<linkage boundary = "bar.htm" >
  <location URI = "http://www.xx.com/app/ev1.evt" />
  <location URI = "lid://www.xx.com/app/ev3.evt" />
</linkage>

```

Where "http://www.xx.com/app/ev1.evt" etc. are the URIs of the behaviour files.

#### 8.7.3.4 Name and location of linkage file

The event linkage file is located in the same directory as the application entry point document.

The file name of the linkage file is formed by replacing any file name extension with the string "lnk". So, for the application entry point document:

```
foo.html
```

the linkage file would be named:

```
foo.lnk
```

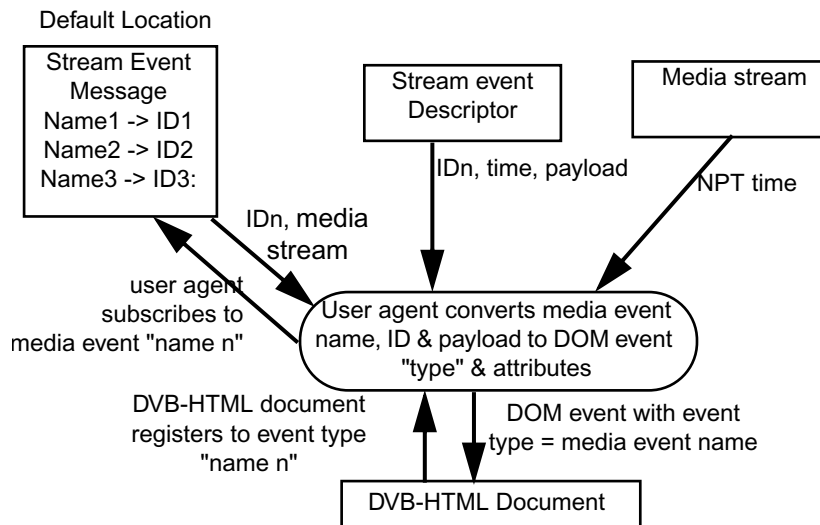
### 8.7.4 Default Trigger Mechanism

The default trigger mechanism is assumed when there is no linkage file associated with a DVB-HTML application. This default mode provides a simple event mechanism in the case where a DVB-HTML application is associated with system events or with trigger events with simple data payload. An overview is provided in figure 12, "[Overview of the default trigger mechanism](#)" on page 102.

In that case, any document of the DVB-HTML application that registers for trigger-events is associated with the default event factory file as defined in 8.7.2.4, "[Default Event Factory File](#)" on page 97. All stream events used in the context of the DVB-HTML application are defined in DSMCC Stream Event messages available in the default location (see 8.7.1.3, "[Binding to DSM-CC Stream events](#)" on page 92). It is assumed that the name values of the stream events are identical to the trigger event types. Stream events with name values different from trigger event types registered by a document or different from system event names shall be ignored.

In the default mode, all DSMCC Stream Event messages present in the default location will be bound to the application. An application that has registered on a given event name will thus receive several events if they are referenced in different DSMCC Stream Events with this exact value of event name. This may not reflect the original intent of the application author. It is thus recommended that application authors carefully choose the values of event type in their DVB-HTML application when they register on stream events that are originating from different media sources (e.g. by using prefixed names).

Event registration by a DVB-HTML document is performed as described in 8.7.1.2, "Application registration and reception" on page 92 except that the listeners need to be positioned on the root element node of the DOM tree, since the target of the event will be this node. An example is provided in 8.7.2.3, "Default Event Factory Element" on page 97.



**Figure 12 : Overview of the default trigger mechanism**

The following text provides an informative description of the different steps involved when a DVB-HTML document subscribes to a trigger event in the default mode:

1. The DVB-HTML document, through an embedded Xlet or any attached script, implements an EventListener interface and subscribes to the trigger event through a call to the DOM `addEventListener()` method on the root element node of the DOM tree with an appropriate event type.
2. The user agent locates the DSMCC Stream Event message in the default location that contains an event name equal to the registered event type.
3. The DSMCC Stream Event message provides the mapping between the event name and the event id and references the source of the stream event descriptors that the user agent needs to monitor.
4. On receipt of the stream event descriptor that provides the mapping between the event id and both the eventNPT and the associated payload, the user agent begins the synthesis of the DOM event, as corresponding to the default event factory element. When the NPT of the referenced stream is equal to the eventNPT of the stream event descriptor or as soon as the stream event descriptor is received in case of 'do-it-now' events, the DOM event is injected in the DOM implementation.
5. Payload of the stream event can be fully extracted from the synthesised DOM event through the default 'payload' attribute.

## 8.8 CSS

Cascading Style Sheets (CSS) are a method to attach a look to HTML or XML content.

### 8.8.1 Summary of CSS profiling for MHP

DVB-HTML shall support CSS2 as defined in [CSS 2 \[87\]](#) but amended by the following profile statements:

- "MHP profile of CSS data types" on page 103
- "MHP profile of CSS @ rules" on page 103
- "MHP profile of CSS media types" on page 104
- "Graphics and video integration" on page 105
- "Font selection" on page 117
- "Font specification" on page 118

An MHP terminal must therefore observe the conformance clauses specified in [CSS 2 \[87\]](#).

### 8.8.2 MHP profile of CSS data types

The implementations shall support all of the data types defined in [CSS 2 \[87\]](#) except as amended by table 19 "MHP profile of CSS data types".

**Table 19 : MHP profile of CSS data types**

Data type	Profile
<alphavalue>	See 8.8.4.2.1.1, "opacity" on page 104 The results of opacity blending may be implementation specific, see 13.6.1, "Approximations in composition" on page 350.
<angle>	Not required (support for aural style sheets is not required).
<color>	Application authors should not make any assumption on the expected results when using system colours defined in <a href="#">CSS 2 [87]</a> . In particular there is no guarantee that any two system colour names will have distinct appearances.
<frequency>	Not required (support for aural style sheets is not required).
<length>	See 8.8.5.2, "Coordinate spaces" on page 106
<shape>	In this specification the only valid <shape> value is:  <pre>rect (&lt;top&gt; &lt;right&gt; &lt;bottom&gt; &lt;left&gt;)</pre> The parameters <top>, <bottom> <right>, and <left> specify offsets from the top and left of the box.
<time>	Not required (support for aural style sheets is not required).

### 8.8.3 MHP profile of CSS @ rules

The implementations shall support all of the @ rules defined in [CSS 2 \[87\]](#) except as amended by table 20 "MHP profile of CSS @ rules".

**Table 20 : MHP profile of CSS @ rules**

Rule	Profile
page	Not required.
viewport	Required, See 8.8.5.3.2, "The @viewport rule" on page 107.

## 8.8.4 MHP profile of CSS media types

A DVB-HTML user agent is required to support the media types in this section, support for other media types is optional.

### 8.8.4.1 "screen" media type

A DVB-HTML user agent shall support the `screen` media type

### 8.8.4.2 'dvb-tv' media type

A DVB-HTML user agent shall support the `dvb-tv` media type.

The `dvb-tv` media type is a new media type introduced to enable content authors to differentiate an MHP terminal with a predominantly TV like environment (low resolution, possible interlace, specific gamut color, limited-scrollability, small screens, sound available) from an MHP terminal with a more computer desktop like environment. The `dvb-tv` media type is a superset of properties as the `screen` media type as defined in [CSS 2 \[87\]](#). `dvb-tv` is a member of the following media groups.

**Table 21 : Relationship between media groups and media types**

MediaTypes	Media Groups			
	continuous/paged	visual/aural/tactile	grid/bitmap	interactive/static
dvb-tv	continuous	visual	bitmap	both

#### 8.8.4.2.1 Additional Properties of "dvb-tv" media type

##### 8.8.4.2.1.1 opacity

**opacity:** Objects may be rendered in a semi-transparent fashion in DVB-HTML using the `opacity` property defined in [SVG \[91\]](#). Opacity can be thought of as a postprocessing operation. Conceptually, after an object or group of objects is rendered into an RGBA off-screen image, the opacity setting specifies how to blend the off-screen image into the graphics device.

Value: <alphavalue> | inherit

Initial: 1.0

Applies to: all elements

Inherited: no

Percentages: N/A

Media: visual

**<alphavalue>:** The uniform opacity setting to be applied across an entire object. Any values outside the range 0,0 (fully transparent) to 1,0 (fully opaque) will be clamped to this range. If the object is a container element such as a `<div>`, then the effect is as if the contents of the `<div>` element were blended against the current background using a mask where the value of each pixel of the mask is `<alphavalue>`.

See alpha compositing in [13.3.4, "Composition" on page 343](#).

The CSS boxes are composed following the normal CSS model, however overlapping boxes that have semi opaque colours are composed using the Porter-Duff rules (see [Porter-Duff \[D\]](#)). We define a new property:

##### 8.8.4.2.1.2 compose-rule

**compose-rule:** The default rule used is the `src-over` rule.

CSS allows for a fully transparent background, and with the addition of the compose rule and semi opaque colours semi transparent effects can be achieved. The various approximations defined for compositing in [13.3.5, "Composition Rules" on page 344](#) also apply to the CSS compositing.

Value: clear | dst-in | dst-out | dst-over | src | src-in | src-out | src-over

Initial: src-over

Applies to: all elements

Inherited: no

Percentages: N/A

Media: visual

#### 8.8.4.2.1.3 clip-video

See 8.8.5.8.1.1, "clip-video" on page 116.

#### 8.8.4.2.1.4 focus traversal

See 8.8.5.10, "Focus traversal and short-cuts" on page 116

#### 8.8.4.2.1.5 viewports

See 8.8.5.3.2, "The @viewport rule" on page 107

### 8.8.4.2.2 Policy rules

When several media types are present in a DVB-HTML document, user agents shall apply the properties settings that are included in the most appropriate @media rule. In this case, the user agent shall ignore properties settings that are included in the other @media rules targeting other media types. An MHP terminal shall ignore all properties belonging to media types that it does not support or are not selected.

### 8.8.4.3 Clarifications on support of paged properties

The CSS2 specification[87] appendix F lists properties associated with the paged media group as being also associated with the visual media group, which creates a possible ambiguity with the definition of the screen and dvb-tv media types which do not support the paged media group. For clarification, DVB-MHP does not require the support of the properties where the "media group" column contains "visual, paged".

## 8.8.5 Graphics and video integration

### 8.8.5.1 General recap of the MHP graphics

The MHP defines three sets of planes, background, background video, and application. The DVB applications draw into the application layer, but may be transparent or translucent to the video planes beneath. Video objects can be placed in the application layer too, and these act as components (e.g. being clipped to the viewport).

MHP defines the following coordinate spaces which can be used in these planes,

#### 8.8.5.1.1 Input video space

This considers post upsampling MPEG pixels.

#### 8.8.5.1.2 Device space

Logical pixels in the various display devices. There may be different device spaces for the various device types (e.g. video and graphics).

#### 8.8.5.1.3 Normalised space

Normalised coordinates relative to the screen.

#### 8.8.5.1.4 Colour

MHP defines a 4 dimensional colour space coordinate system, which covers the sRGB gamut with semi opaque colours.

In DVB-HTML points in the three chroma dimensions are specified using the CSS `rgb()` or `#rgb` syntax, the `opacity` property is used to specify the fourth translucency component (see 8.8.4.2.1.1, "opacity" on page 104). Any translucency information in referenced images shall also be respected.

## 8.8.5.2 Coordinate spaces

DVB-HTML applications do not have to fill the entire graphic space, but can specify a rectangle in the screen space, to locate the initial viewport for the application. It can also specify and use a local pixel coordinate space in that viewport for placement of elements within the application.

DVB-HTML applications can also make use of locations in the video space so that elements can be positioned relative to them (e.g. placing captions in black bars).

The definition of `em` and `ex` shall be computed using the rules in annex D.3.4.2, "Horizontal resolution" on page 426, not based on the subtended angle at the eye as in CSS 2.

### 8.8.5.2.1 Screen coordinates

The screen space coordinates are normalised, and can therefore be expressed in CSS % units where it is defined for MHP that this means percent of screen real estate.

i.e. full screen is `top:0% left:0% width:100% height:100%`.

### 8.8.5.2.2 Pixel coordinates

Pixel coordinates use CSS `px` units, these describe logical pixels in the viewport (which in the same way as `HGraphicsDevice`, may not actually map to physical device pixels), and define the coordinate space of the viewport.

### 8.8.5.2.3 Video coordinates

To address positions based on the video coordinate space the `pel` numeric type is defined.

e.g. the style

```
#captions { position: fixed; top: 23pel; left: 43pel; width: 100pel; height: 60pel }
```

establishes a fixed box in the viewport, the rectangle of which is based on (43, 23, 100, 60) in the source video (see "Definition of pel areas in the video" on page 115 for how this is computed into an actual box).

### 8.8.5.3 How to define the initial containing block

#### 8.8.5.3.1 Problem

The rectangle on screen that is the logical equivalent of the `HScene` in DVB-J, is called the *viewport* in CSS. Since this does not have to fill the screen, it is required to be able to establish the coordinate space that this viewport has (if this is not specified by the application then the user agent can use the natural pixel resolution for the device). Within the viewport the user agent needs to establish the root containing block for the document to be laid out in - following the normal CSS rules, this is defined in the viewport pixel coordinate space and may not extend to the full size of the viewport.

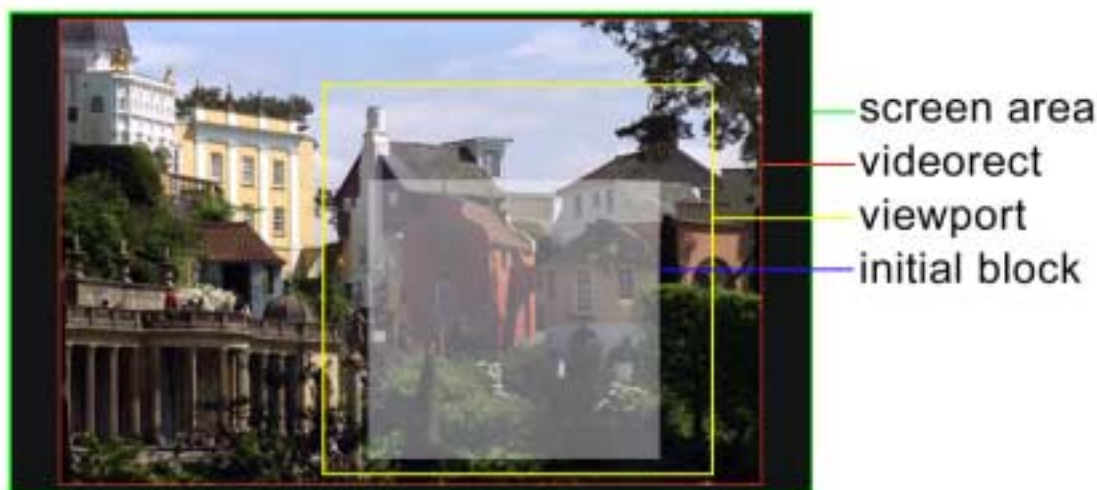


Figure 13 : CSS Boxes

#### 8.8.5.3.2 The @viewport rule

**Viewport context:** The `@viewport` rule is used to define the viewport for the application. This is roughly equivalent to the `@page` rule for printed output. The declarations that it contains are defined to be in the **viewport context**.

Only one viewport can be in effect at a time, and this is defined by the cascade referenced by the stylesheet of the root containing document (see 8.8.5.4, "Cascading" on page 113 and also 8.8.5.3.4, "Pseudo classes" on page 112) for example:

```
@viewport {
  scene: (25%, 25%, 50%, 50%);
  hres: 288px;
  vres: 360px;
  initial: (94px, 130px, 100px, 100px)
}
```

Specifies that:

- the viewport's scene is 1/4 screen and centred,
- that the desired coordinate resolution of the viewport is 288x360,
- and the initial containing block is a 100x100 square approximately centred within the viewport.

### 8.8.5.3.3 Establishing a viewport

#### 8.8.5.3.3.1 Viewport properties

The following properties may be used inside a `@viewport` rule. DOM access to these properties is provided by 8.8.5.9, "DOM Access to CSS" on page 116. Implementations are not required to respect property requests in the `@viewport` rule that would require the implementation to exceed the minimum requirements defined in G.3, "Video" on page 445. By giving DOM write access to these properties the application can dynamically change the backgrounds. By giving DOM read access the actual values can be discovered to adapt to the current local conditions.

**scene:** The `scene` property specifies where the CSS viewport lies in terms of percentage of the area allocated to the application (this area is by default based on the video rectangle but may be changed, see 8.8.5.5.1, "The area property" on page 113).

Value: `<shape>`

Initial: `rect(0%, 100%, 100%, 0%)`

Applies to: [Viewport context](#)

Inherited: no

Percentages: area

Media: visual

**<shape>:** See table 19, "MHP profile of CSS data types" on page 103. In this context `<top>`, `<bottom>`, `<right>`, and `<left>` specify `<percentage>`.

**<percentage>:** The offset is a percentage of the width (for `left` or `right`) or height (for `top` and `bottom`) of the area (see 8.8.5.5.1, "The area property" on page 113).

**horizontal-resolution:** Determines the number of logical pixels horizontally in the viewport and defines the grid to which `px` units in the stylesheet apply. NB the logical pixels do not have to match with the actual device pixels.

Value: `<length>`

Initial: 720px

Applies to: [Viewport context](#)

Inherited: no

Percentages: N/A

Media: visual

**vertical-resolution:** Determine the number of logical pixels vertically in the viewport and defines the grid to which `px` units in the stylesheet apply. NB the logical pixels do not have to match with the actual device pixels.

Value: `<length>`

Initial: 576px

Applies to: [Viewport context](#)

Inherited: no

Percentages: N/A

Media: visual

**initial:** This establishes the box for the root element of the layout. CSS allows that the initial containing block be larger than the viewport, and in this case it states the user agent should offer a mechanism for scrolling, since this may not be appropriate on an MHP it is advised that authors do not use initial blocks larger than the viewport. If they do then the user agent shall clip to the viewport, and may provide mechanisms to view the clipped regions.



Value: <shape>

Initial: rect(0%, 100%, 100%, 0%)

Applies to: [Viewport context](#)

Inherited: no

Percentages: Scene

Media: visual

**<shape>**: See table 19, "MHP profile of CSS data types" on page 103. In this context <top>, <bottom> <right>, and <left> specify <length> or <percentage>.

**<length>**: The offset is a fixed distance from the reference edge (top or left) in the logical pixels of the scene.

**<percentage>**: The offset is a percentage of the width (for left or right) or height (for top and bottom) of the viewport established by the scene property.

#### 8.8.5.3.3.2 Other viewport properties

These properties will mostly be useful for interrogation of the video processing being performed so that a presentation can be tailored to fit, but they also allow the manipulation of the background devices.

These are used after the viewport is selected (see "Pseudo classes" on page 112), and allow the application to override the presentation of the background properties.

**type**: The type defines whether the viewport should attempt to preserve graphic fidelity or video fidelity. It can also require that the pixel coordinate space be aligned with the video coordinate space.

Value: video | graphics | aligned | none

Initial: none

Applies to: [Viewport context](#)

Inherited: no

Percentages: N/A

Media: visual

**background-image-rectangle**: This property of the viewport defines the shape of the image placed on the background device.

Value: <shape>

Initial: rect(0%, 100%, 100%, 0%)

Applies to: [Viewport context](#)

Inherited: no

Percentages: screen

Media: visual

**<shape>**: See table 19, "MHP profile of CSS data types" on page 103. In this context <top>, <bottom> <right>, and <left> specify <percentage>.

**<percentage>**: The offset is a percentage of the screen width (for left or right) or height (for top and bottom).

**background**: This property of the viewport defines the image or colour to be used in the background plane when the application has the focus.

Value: <uri> [, <colour>] | <colour>

Initial: black

Applies to: [Viewport context](#)

Inherited: no

Percentages: N/A

Media: visual

If a <uri> is specified it should point to an image in an MHP supported format, other formats may be supported. If the resource is not found then the optional fallback colour or the initial value will be used

**background-video-rectangle-n:** This property of the viewport defines the output rectangle (in percent of the screen) of the background video device (see 8.8.5.3.3.3, "n planes" on page 112 for explanation of the "-n" notation). The default is full screen. The pel aspect ratio of this rectangle is the same as that of the screen. the resolution however is a subset of the screen resolution.

Value: <shape>

Initial: rect(0%, 100%, 100%, 0%)

Applies to: [Viewport context](#)

Inherited: no

Percentages: screen

Media: visual

**<shape>:** See table 19, "MHP profile of CSS data types" on page 103. In this context <top>, <bottom> <right>, and <left> specify <percentage>.

**<percentage>:** The offset is a percentage of the screen width (for left or right) or height (for top and bottom).

#### **background-video-clip-n:**

This property of the viewport defines the input (clipping) rectangle of the background video source (see 8.8.5.3.3.3, "n planes" on page 112 for explanation of the "-n" notation). The clipping maintains the pel aspect ratio of the screen. This clipping rectangle then induces a video transform, which should not be specified. The transform in this case may not be limited to one of the given constants (see "background-video-transform-n:" on page 111), but will usually be "DFC\_PROCESSING\_CCO" on page 111

Value: <shape>

Initial: rect(0%, 100%, 100%, 0%)

Applies to: [Viewport context](#)

Inherited: no

Percentages: input video

Media: visual

**<shape>:** See table 19, "MHP profile of CSS data types" on page 103. In this context <top>, <bottom> <right>, and <left> specify <pels> or <percentage>.

**<pels>:** The offset is specified in the pels of the input video.

**<percentage>:** The offset is a percentage of the input video width (for "left" or "right") or height (for "top" and "bottom").

**background-video-preserve-aspect-n:** This property of the viewport defines how the input video source rectangle is transformed onto the display output rectangle (see 8.8.5.3.3.3, "n planes" on page 112 for explanation of the "-n" notation). If the value is false the video is transformed with no regard for the aspect ratio of the source or destination but simply scales the input grid to the output grid, if true (the default) the aspect ratio of the source is preserved onto the aspect ratio of the destination and scaled as much as possible (some of the output rectangle may be filled with black letterbox bars).

Value: true | false

Initial: true

Applies to: [Viewport context](#)

Inherited: no

Percentages: N/A

Media: visual

**background-video-n:** This property of the viewport defines the locator of the video source or service (See 8.6.18, "DVB Service styling" on page 91) to be used in the background video plane when the application has the focus (see 8.8.5.3.3.3, "n planes" on page 112 for explanation of the "-n" notation). If not specified the MHP uses the media components of the currently playing service.

Value: <uri> (, <uri>)\* | <color>

Initial: dvb.current

Applies to: [Viewport context](#)

Inherited: no

Percentages: N/A

Media: visual

If a <uri> is specified it should point to a video component or service. If the resource is not found then the optional fallback <uri>s may be tried in order, if no resource is found the MHP uses the media components of the currently playing service.

**background-video-transform-n:** This property of the viewport defines the transform of the video to the output rectangle (see 8.8.5.3.3.3, "n planes" on page 112 for explanation of the "-n" notation). It has one of the symbolic constant values:

Value: <process>

Initial: DFC\_PROCESSING\_DEFAULT

Applies to: [Viewport context](#)

Inherited: no

Percentages: N/A

Media: visual

**<process>:** One of the following constant values:

**DFC\_PROCESSING\_CCO:** A central rectangle is cut out of the video input frame and transferred into the output video rectangle defined by the corresponding background-video-rect

**DFC\_PROCESSING\_FULL:** The full video input frame is transferred, part of the output may be black if the video input aspect is not the same as the output frame aspect.

**DFC\_PROCESSING\_LB\_14\_9:** The video input frame is transferred into a 14:9 letterbox in the output frame defined by the corresponding background-video-rect.

**DFC\_PROCESSING\_LB\_16\_9:** The video input frame is transferred into a 16:9 letterbox in the output frame defined by the corresponding background-video-rect.

**DFC\_PROCESSING\_LB\_2\_21\_1:** The video input frame is transferred into a 2,21:1 letterbox in the output frame defined by the corresponding background-video-rect.

**DFC\_PROCESSING\_DEFAULT:** The default decoder format conversion is active.

**DFC\_PROCESSING\_PAN\_SCAN:** An mxn part out of the video input frame is transferred into the output frame defined by the corresponding background-video-rect. The position of this part is determined by pan & scan vectors from the MPEG video stream.

**DFC\_PROCESSING\_UNKNOWN:** Constant representing an unknown format conversion being performed by the decoder. (e.g. if a manual transform is defined). This value cannot be set

#### 8.8.5.3.3 n planes

The "n" is to be replaced by an integer number greater than or equal to zero defining the plane in the video layers, where 0 is the furthest plane from the viewer. The "-n" is optional with background-video-rect equivalent to background-video-rect-0.

#### 8.8.5.3.4 Pseudo classes

Viewports can be labelled with pseudo-classes which allow the application to adapt to the physical structure of the MHP display or output system. For example:

```
@viewport :16x9 { ... }
@viewport :4x3 { ... }
```

The pseudo-class syntax is as follows:

```
display-pseudo-class = ":" [ aspect-ratio ] [ resolution ] [ scan ]
aspect-ratio = integer "x" integer
resolution = "R" integer "x" integer
scan = "P" | "I"
```

integer is an <integer> as defined in the CSS 2 specification

where the components of the pseudo-class have the following meanings:

**aspect-ratio:** specifies the physical display pixel aspect ratio

**resolution:** specifies the physical display pixel resolution

**scan:** specifies whether the display is progressive scan ("P") or interlaced ("I")

This allows most of the common formats to be defined.

A pseudo-class matches a display if each component of the pseudo-class either exactly matches the display or is not present.

EXAMPLE 1: 16x9R720x576P

Matches a progressive scan 16x9 display with 720x576 pixels.

EXAMPLE 2: 4x3R1024x768

Matches a 1024x768 display with 4x3 pixel aspect ratio, either interlaced or progressive scan.

If the pseudo-class for more than one @viewport rule matches a display, one rule is chosen by applying the CSS 2 cascade algorithm ([87] section 6.4 "The cascade"). In addition to the description of specificity there, the relative specificities of several display pseudo-classes is determined by summing the number of components which are specified; for example, "P" has a specificity of 1, "16x9R720x576P" has a specificity of 3.

### 8.8.5.4 Cascading

The @viewport properties cascade in the normal manner. However, only those viewport properties that are defined in the context of the document of the DVB-HTML application displayed in a root window shall be used to determine the application's viewport. Any @viewport properties defined in the context of a document that is not displayed in the root window shall be ignored.

NOTE: Therefore applications running in sub frames cannot control the viewport or background video.

### 8.8.5.5 How to discover where the video is

In order to have good visual synchronization with video in the background plane, a DVB-HTML application will need to be able to determine the intersection of the viewport and the background video areas, both the active video and any additional "bar" areas caused by letterboxing.

#### 8.8.5.5.1 The area property

DVB-HTML defines an @viewport property *area* which defines the rectangle of which the viewport is a sub rectangle. This has the following logical values, which can be combined together.

**screen:** uses the screen as reference, not the background video, this cannot be combined with other values.

**total-video-area-on-screen:** defines the box on screen where the video is including any bars the MHP knows about

**active-video-area-on-screen:** defines the box on screen where the video is excluding bars

**LV-bar:** limits the area to the left hand vertical letterbox bar.

**RV-bar:** limits the area to the right hand vertical letterbox bar.

**TH-bar:** limits the area to the top horizontal letterbox bar.

**BH-bar:** limits the area to the bottom horizontal letterbox bar.

**EXAMPLE 1:** `area: totalVideoAreaOnScreen`

is the default, and would limit the viewport to exactly where the video is being placed (including letterbox bars)

Combinations must add up to a single rectangle.

**EXAMPLE 2:** `area: activeVideoAreaOnScreen & BHBar`

would be legal and would allow the use of the area in the lower letterbox bar but not the top bar

**EXAMPLE 3:** `area: LVBar & THBar`

would be illegal

**EXAMPLE 4:** `area: THBar & BHBar`

would be illegal

**EXAMPLE 5:** `area: THBar & BHBar & activeVideoAreaOnScreen`

would be legal and would allow the use of the area in the upper and lower letterbox bar and the active video area

The order of property evaluation is as follows:

- first the background video rectangle is established (see "[background-video-rectangle-n](#)" on page 110, the default is full screen). The viewport is established with respect to the furthest video plane from the viewer.
- then the area is determined, (the default is the total active video area). and finally the viewport and initial box are determined. These would all have to be recalculated if the background video rectangle were to change (either by the terminal or by DOM control)

### 8.8.5.6 Placing content in relation to video

The terminal defined starting cascade style sheet implicitly defines (see "[Definition of boxes](#)" on page 115) the following regions when the viewport contains them.

```
.active-video-area-on-screen
.total-video-area-on-screen
.LV-bar
.RV-bar
.TH-bar
.BH-bar
```

so that for example the document

```
<!DOCTYPE
  html PUBLIC "-//DVB//DTD XHTML DVB HTML 1.0//EN"
  "http://www.dvb.org/mhp/dtd/dvbhtml-1-0.dtd"
>
<html>
<head>
  <title>css demo</title>
  <style type="text/css">
  <!--
  @viewport 16x9I{
    area: active-video-area-on-screen & BH-bar
  }
  -->
</style>
</head>
  </body>
    <div class="active-video-area-on-screen"> </div>
    <div class="BH-bar"> </div>
  </body>
</html>
```

Would use allow the placement of material in the lower black bar and over the video area.

### 8.8.5.6.1 Definition of boxes

The named regions are defined as the intersection of the viewport (shown in the illustration as the yellow rectangle) with the relevant area, therefore the regions could potentially be empty, in this case no content is displayed for that region. For example the `#TH-bar` shown in blue in the illustration below does not cover the whole of the letterbox bar, but only that which intersects the viewport. Similarly the `active-video-area-on-screen` region shown by the purple rectangle does not cover all the live video, but only that which intersects the viewport.



Figure 14 : Example of named screen regions

### 8.8.5.6.2 Definition of pel areas in the video

Positions defined by using the pel type are defined to be the pixel location in the viewport where the corresponding video pel is translated to. In the case that the transformed pel covers more than one pixel, the top most left most pixel is used. This allows the placement of content in relation to elements within the video.

Note: the pixel may not be within the viewport, and so may get clipped. it could also change with pan and scan.

### 8.8.5.7 Placing video within the presentation

#### 8.8.5.8 Box Layout

##### 8.8.5.8.1 Video Boxes

Video content may be placed in an `<object>` or `<img>` element. Using style rules the box for this can be positioned following the normal CSS rules anywhere within the viewport. This acts like a video component in DVB-J, and is entirely separate from video in the background plane.

When placing video inside an element, the video inside that element is positioned and scaled by any positioning and resizing of the element due to style. If the implementation does not support arbitrary scaling of video, then the resource shall be treated as not available and the fallback elements (if any) applied, otherwise the video shall be scaled as required. The video is always scaled to the full size of the element.

NOTE: Since the video is scaled to exactly fit the elements box, the `overflow` and `clip` attributes have no meaning for such elements.

Video element objects are treated as defined by the styling of the element. However, it is not required to support `opacity` style on such elements.

### 8.8.5.8.1.1 clip-video

The mapping to the source video is provided by the `clip-video` property which allows an arbitrary portion of the video to be placed within the element.

**clip-video:** The `clip-video` property defines the bounding box of the source rectangle in the source video allowing an arbitrary portion of the video to be placed within the element (as in figure 38, "Introducing video into the AWT component stack" on page 348). The coordinate space used to express the region is that of the decoded video after possible TR 101 154 [9] up-sampling. This rectangular segment of the video is then used to fill the element to which the style property is applied

For example:

```
<object src="dvb://current" style="clip-video:rect (10pels, 50pels, 70pels, 90pels)"/>
```

Value: <shape>

Initial: full video rectangle

Applies to: object, image

Inherited: no

Percentages: N/A

Media: visual

Media: visual

**<shape>**: See table 19, "MHP profile of CSS data types" on page 103. In this context <top>, <bottom> <right>, and <left> specify <pels>

**<pels>**: The offset is specified in the pels of the input video and indicates an offsets from the left or top edge of the video as appropriate.

### 8.8.5.9 DOM Access to CSS

In order to gain programmatic access to the `@viewport` rule the language specific binding for the DOM interface specified in 8.11.8.1.3, "DVBCSSViewportRule" on page 158 may be used.

### 8.8.5.10 Focus traversal and short-cuts

In MHP it is possible to determine the order of traversal of elements using the navigation keys, and to specify certain keys as short cuts.

The `dvb-tv` media type includes the following additional style properties:

**Table 22 : Navigation properties**

Property	Meaning
nav-up	References the visual element to receive focus on receipt of the VK_UP event
nav-down	References the visual element to receive focus on receipt of the VK_DOWN event
nav-left	References the visual element to receive focus on receipt of the VK_LEFT event
nav-right	References the visual element to receive focus on receipt of the VK_RIGHT event
nav-index	Provides a navigation number for the element. This number must be unique on the page.
nav-first	An element with this style will be given focus on first presentation of the page.



The values of `nav-up`, `nav-down`, `nav-left`, and `nav-right` may be `<integer>`, in which case the referenced element is the element with the corresponding `nav-index` style, or they may be `<string>`. The `<string>` has the following syntax:

```
value:      '[outer]' | [frame-ref] '#' [elem-ref]
frame-ref:  frameid |
elem-ref:   '$'integer | elemid
```

where:

**frameid:** is the id of the frame containing the referenced item (the current frame is the default). The same rules shall apply as for the target attribute in [HTML 4 \[90\]](#), except that steps 3 and 4 of Annex B.8 therein shall not apply as DVB-HTML does not open new windows.

**elemid:** is the id of the referenced item. At least one of the `frame-ref` or `elem-ref` optional elements must be present if value is not `[outer]`.

**integer:** is a positive integer, and represents the element with the corresponding `nav-index` style (the default is the element with the `nav-first` style).

e.g.

```
<style type="text/css">
  #foo nav-right:"frame1#bar"
</style>
```

Would indicate the style for the element with the id `foo`, would navigate the focus to the element with the id `"bar"` in the frame `"frame1"`.

The following string is reserved and has special meaning:

**[outer]:** This string refers to the external application context in the case that the DVB-HTML application is an inner application. If this name is used in the case where the application is not an inner application it shall be ignored.

If the element to receive the navigation focus cannot be determined by these rules, then the user agents default focus processing should be applied.

NOTE: The navigation hints should be used with some care, as the user agent may have applied defaults based on the current layout. If an author overrides these defaults, then the layout may produce non intuitive results for the viewer.

## 8.8.6 Font selection

There are four possible font selection actions defined in CSS2: name matching, intelligent matching, synthesis, and download. A user agent shall support the font matching algorithm as defined in section 15.5 in [CSS 2 \[87\]](#). The user agent is not required to support intelligent font matching algorithm and font synthesis or their associated descriptors as described in "[Descriptor support in MHP](#)" on page 117. The user agent shall support the CSS2 `@font-face` rule and shall ignore any font index file co-located with a DVB-HTML application.

**Table 23 : Descriptor support in MHP (Sheet 1 of 2)**

Descriptor	Required
ascent	
baseline	
bbox	
cap-height	
centerline	
definition-src	
descent	
font-family	Yes

**Table 23 : Descriptor support in MHP (Sheet 2 of 2)**

Descriptor	Required
font-size	Yes
font-stretch	Yes
font-style	Yes
font-variant	Yes
font-weight	Yes
mathline	
panose-1	
slope	
src	Yes
stemh	
stemv	
topline	
unicode-range	Yes
units-per-em	Yes
widths	
x-height	

### 8.8.6.1 Restrictions on "src" descriptor

The user agent shall fully support the descriptor for Referencing: `src` with the following restrictions:

- The only format type required to be supported for the font `src` descriptor is "truedoc-pfr".
- The only string required to be supported for the `<font-face-name>` value for access to local fonts is "Tiresias", see G.4.1, "The built-in font" on page 445.

### 8.8.7 Font specification

The user agent is required to support the following set of properties for selecting a font:

- `font-family`
- `font-style`
- `font-variant`
- `font-weight`
- `font-stretch`
- `font-size`
- `font-size-adjust`
- `font`

Setting `font-stretch` and `font-size-adjust` is not required to have any visual effect in a constrained graphics environment.

The text rendering model used by the user agent shall respect the text rendering model defined in D.3.4, D.3.5, D.3.6.1 and D.3.6.2.

### 8.8.8 Default behaviour

In the absence of any other styling information for a page, the user agent shall render using the default stylesheet in Annexe AB, "(normative): DVB HTML StyleSheet" on page 1280

### 8.8.8.1 Default style sheet font rules

For DVB HTML user agents the style sheet referred to by CSS 2 (see UA default stylesheet in [CSS 2 \[87\]](#)) will consist of the stylesheet in appendix [AB on page 1280](#). This stylesheet is not directly accessible to DVB HTML applications.

In addition to the style entries listed in the appendix, this stylesheet shall also provide `@font-face` rules for all installed fonts, which shall at least cover the minimum installed set of fonts (see [G.4, "Resident fonts and text rendering" on page 445](#)) and have rules for the CSS 2 generic families, `serif`, `sans-serif`, `cursive`, `fantasy` and `monospace`.

The minimum requirement of the additional font rules are:

- The implementation-specific stylesheet shall provide a font description with name "Tiresias".
- The implementation-specific stylesheet shall provide the normal weight.
- The implementation-specific stylesheet shall provide the normal stretch.
- The implementation-specific stylesheet shall provide at least the point sizes required (see table [G.2, "Minimum set of sizes" on page 445](#)).
- The implementation-specific stylesheet shall provide a `src`: to locate the font data.
- The implementation-specific stylesheet shall not provide `font-style` or `font-variant` declarations unless the implementation also provides font data that would visually distinguish the rendered text.

For example a minimal implementation might use the following rule:

```
@font-face {
  font-family: "Tiresias", serif, sans-serif, cursive, fantasy, monospace;
  font-weight: normal;
  font-stretch: normal;
  font-size: 36pt, 31pt, 26pt, 24pt;
}
```

#### 8.8.8.1.1 Extending the simple rule

An implementation may additionally include `unicode-range`: data for the installed fonts, and other "Descriptors for Matching" ([CSS 2 \[87\]](#) spec Section 15.3.6), which might be used, e.g. for `font-size-adjust`.

#### 8.8.8.1.2 Fallback for italic, small caps and font stretch

The rules above imply that the available fonts cover all variations of `italic`, `small-caps` and `font-stretch`. An implementation that only had the minimum installed fonts should present the same (plain) style glyphs for any combination of the styles required by CSS 2.

NOTE: This allows an author to use `font-style: italic`, `font-weight: bold` etc. in stylesheets without having text so styled appear as missing character glyphs when using Tiresias, but they should be aware that no extra styling information may be displayed.

## 8.9 Xlet integration

DVB-HTML defines an `<object>` element that may be used for embedded Xlets. For `<object>` elements used for embedded Xlets the type should be set to "application/dvbj", the `<object>` element may contain `<param>` elements containing run time arguments for the Xlet defined by the `<object>` element. The sequence of these arguments is mapped to `XletContext.ARGs` and shall be available to the Xlet, in the form of an array of Strings, by calling:

```
javax.tv.xlet.XletContext.getXletProperty(XletContext.ARGs)
```

If no valid arguments are defined, this method shall return an array of strings with zero elements.

By default the application identifier of Xlets contained in DVB-HTML are the same as that of the enclosing application.

Using the `AppID` attribute the embedding HTML context can supply a different `AppID` for the Xlet, thus giving it a different context (e.g for persistent file locations). The value given for the `AppID` attribute must be one of those given in the [DVB-HTML application descriptor](#) (see 10.10.1 on page 231). An Xlet with an `AppID` assigned by the HTML application is still considered part of the single DVB HTML application, e.g. for inter application communication.

If an Xlet and an embedded Xlet have the same application identifier it is only possible for one to be running at any time. So, if the Xlet has already started then a DVB-HTML application shall not be able to start an embedded Xlet with the same application identifier. Conversely, if the embedded Xlet has already started the Xlet shall not start. Any lifecycle signalling associated with an Xlet with the same application identifier as an embedded Xlet shall not apply to the embedded Xlet. If DVB-HTML application attempts to start an embedded Xlet that has a an application identifier that conflicts with an already running application:

- the embedded Xlet shall not be started
- the already running application is not considered apart of the DVB-HTML application

### 8.9.1 Object element

When used to include an Xlet inner application the following object element attributes semantics shall apply:

**declare:** when present, makes the current object definition a declaration only. The object must be instantiated by a subsequent object definition referring to this declaration.

**classid:** specifies the location of the class that implements the interface `javax.tv.xlet.Xlet`. Exactly one instance of this Xlet class shall be created. The URI shall be evaluated as relative to the `codebase` attribute.

**codebase:** specifies the base URI for the Xlet. Xlet classes are loaded relative to the "directory" specified by this locator. If this attribute is not specified, then it defaults to the same base URI as for the DVB-HTML document carrying the object element. If the URI is not within the application containing the Xlet then the Xlet shall not be loaded.

**codetype:** specifies the content type of the data referenced by the `classid` and `codebase` attributes. In the case of Xlets it will be set to the MIME type for Xlets (`application/javatv-xlet`).

**archive:** This is ignored for embedded Xlets.

**standby:** specifies a message that a user agent may render while loading the objects implementation and data.

**height, width:** these attributes specify the size of the Xlet's visible representation. If either height or width is zero or negative, then the method `javax.tv.graphics.TVContainer.getRootContainer()` shall return null.

## 8.9.2 Param element

The param element is defined for embedding inner application Xlets as follows:

```
<!ELEMENT param EMPTY>
<!ATTLIST param
  id          ID          #IMPLIED
  name        CDATA       #REQUIRED
  value       CDATA       #IMPLIED
  valuetype   (data|ref|object) "data"
  type        CDATA       #IMPLIED
>
```

where the specified attributes are defined as follows:

**id:** is a unique identifier for the param element. The scope of uniqueness is the document carrying the param element.

**name:** should be set to either `arg_n` (case-insensitive), where `n` is a non negative integer, or, `appid`.

- Where the name is of the form `arg_n` this specifies that the value argument contains the value of run time argument number `n` for the Xlet. If the largest value for `n` among the param tags is `x`, then the ARGV array shall contain `x+1` elements. If two param elements with the same name are given for an Xlet, the corresponding array element shall contain the value of one of them. Which one is chosen is not specified.

These arguments are available to an Xlet through `XletContext.getXletProperty( XletContext.ARGV )` as defined in [Java TV \[51\]](#).

- The `appid` shall carry a 16 bit bit string (left bit first) encoded as follows:

```
appid = "0x" 4*hex
hex   = digit | "A" | "B" | "C" | "D" | "E" | "F" | "a" | "b" | "c" | "d" | "e" | "f"
digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

If more than one parameter tag with the name `appid` is provided or the value does not correspond to that of one of the application ids listed in the [DVB-HTML application descriptor](#) (see [10.10.1 on page 231](#)) the Xlet shall not be started.

**value:** is the string value of this run time argument.

**valuetype:** ignored. Xlet arguments can only be strings, so it doesn't make sense to provide values of any other type.

**type:** ignored. This follows from the treatment of `valuetype`.

## 8.9.3 Example

The following is an example of using object and param elements in order to include an Xlet inner application in a DVB-HTML document:

```
<object
  type = "application/dvbj"
  id = "slideShow"
  codetype = "application/javatv-xlet"
  classid = "slideShow.class"
  codebase = "first_xlet/"
  height = "180"
  width = "320">
  <param name = "arg_0" value = "Everest"/>
  <param name = "arg_1" value = "Kilimanjaro"/>
  <param name = "appid" value = "0xbdcd"/>
</object>
```

## 8.10 Scripting

ECMAScript is a simple lightweight object-oriented scripting language for manipulating computational objects within a host environment. This specification requires ECMAScript as defined in [ECMAScript \[81\]](#).

### 8.10.1 DOM 2 binding

This specification requires the DOM 2 binding libraries for the interfaces required in "[Document Object Model \(DOM\)](#)" on page 128 as defined in the respective references.

### 8.10.2 Interface between ECMAScript and DVB-J

By allowing DVB-J to expose APIs to ECMAScript, applications may be created which are a hybrid of DVB-J and DVB-HTML/ECMAScript. For example, the core engine for a particular application could be implemented in DVB-J and the user interface and graphical content implemented in DVB-HTML/ECMAScript.

#### 8.10.2.1 ECMAScript APIs for accessing DVB-J

Public DVB-J: packages, classes, methods and fields shall be visible in ECMAScript using a property of the global object called Packages. For example, RMI's Naming class would be accessed using Packages.java.rmi.Naming. The java.lang package is accessed using Packages.java.lang. In addition to DVB-J classes, class files located in the codebase as specified by the meta tag described below are also accessible.

Xlets may be accessed via the DOM or accessed via the inter application communication API, see 11.7.3, "[Inter-Application and Inter-Xlet communication API](#)" on page 263.

When ECMAScript uses the Packages object to reference a Java class that is not loaded, the codebase is searched for the class file. The codebase is specified using a meta element with a name "codebase" and the "content" specifying the classpath as a base URL, e.g.

```
<meta name="codebase" content="http://mytv.com/sports/">
```

If multiple codebase meta elements are present in the root document, each specified codebase is searched in its order of appearance. If the class is not found, a java.lang.ClassNotFoundException is thrown. Only meta elements on the root document of the DVB-HTML application may specify the codebase; others have no effect.

ECMAScript contexts within an application share a single classloader. This classloader is separate from that of any Xlets that are part of the DVB HTML application.

#### 8.10.2.2 Inter-Xlet and Xlet-ECMAScript Communication via org.dvb.ixc

In MHP, Xlets are allowed to communicate with each other by exporting objects via the org.dvb.io.ixc APIs described in 11.7.3, "[Inter-Application and Inter-Xlet communication API](#)" on page 263 and annex Y, "[\(normative\): Inter-application and Inter-Xlet communication API](#)" on page 1261. When an Xlet wishes to export an object so that it is visible to other applications, it is exported under a name formed from a string that contains a unique application identifier, possibly with an arbitrary name appended.

With Xlets embedded in a DVB-HTML document, there is the possibility that there will be multiple active Xlets that are part of the same application. Each such Xlet will have its own classloader (see 8.10.2.1 "[ECMAScript APIs for accessing DVB-J](#)"), and will not be able to directly share instances with other Xlets. The Xlets will be able to communicate with each other, using the inter-xlet communication mechanism, accessed through the org.dvb.io.ixc APIs.

The MHP naming convention for exported objects are preserved. The APIs provide two options for exporting objects: Exporting them so that they are only visible within the same application, or exporting them so that they are visible to other applications. In either case, it is possible for two Xlets to export an object under the same name. If this happens, the last object exported shall be the one that is visible; an export from one Xlet might therefore overwrite an export from another that occurred earlier in time. Note that Xlets that are in the same DVB-HTML document (and part of the same application) are expected to be authored to work together, so it is the application author's responsibility to ensure that no harmful overwriting occurs.

### 8.10.2.3 Security

The DVB-J security model applies to the application as a whole, hence an Xlet embedded in an DVB-HTML document has the privileges of the overall DVB-HTML application. ECMAScript may directly invoke DVB-J with the same permissions as the overall application.

See also 8.14, "Security of DVB-HTML applications" on page 164.

NOTE: ECMAScript can pass both internal and external strings across the bridge to Java, but Java has no way to distinguish between them. Hence, Java code written for use across the bridge should treat such strings as it would any other strings from untrusted sources.

### 8.10.2.4 Implicit Method Selection

When more than one method signature matches the set of arguments, argument preference is compared for each argument starting with the first, until one signature has a preferred match. Preference is determined from the argument conversion tables in the following section. In those tables, type conversions are listed in decreasing order of preference.

### 8.10.2.5 Explicit Method Selection

Methods may be selected explicitly by referring to a method as if it were an element of an ECMAScript associative array indexed by the signature, e.g.

```
new Packages["java.lang.String"]["(char[])"] (c);
```

### 8.10.2.6 Static Method Invocation

Static methods of a Java class shall be invocable from ECMAScript either on the JavaClass object or on an instance of the class.

### 8.10.2.7 Method Signature Matching

Typically, methods invoked from ECMAScript are designed for that purpose and unambiguous. In most cases, it is probably sufficient to leave method selection up to the implementation, however the tables below define both the preferred order used for selecting signatures and the type conversion invoked when an argument is passed.

Method selection is performed similarly to DVB-J. The DVB-J method must:

- be public
- be static or non-static depending on whether the ECMAScript invocation is on the class or on an object, respectively.
- have the same number of arguments
- have arguments with types to which the ECMAScript arguments can be converted according to the tables below.

If these conditions are not met, the [DVBException](#) with the error code [CONVERSION\\_TYPE\\_ERR](#) is raised.

If there is only one DVB-J method matching these criteria, that method is invoked.

If more than one DVB-J method matches, then a method is preferred over another for invocation when the conversion of at least one of its arguments is preferred and the conversion of all other arguments is equivalent or preferred. In cases where no method is preferred over all others, results may be implementation dependent.

### 8.10.2.8 New ECMAScript Object Types

ECMAScript is a loosely typed language with few fundamental types (mainly: Number, Boolean, String, Object, Null and Undefined). In order to access DVB-J's richer type structure, three new ECMAScript object types are introduced to support values returned from calls to DVB-J.

**Table 24 : New ECMAScript Object Types**

ECMAScript Object Type	Description
JavaArray	Wrapper for a DVB-J array. Behaves like an ECMAScript array including being indexed by an integer and having a length property
JavaObject	Wrapper for a DVB-J Object. Converting this wrapper to a string, calls the toString() method. Converting to a number calls the doubleValue() method. Converting to a boolean results in true, unless the object is null.
JavaClass	Wrapper for a DVB-J class

### 8.10.2.9 Type Conversion (ECMAScript to DVB-J)

In the following tables conversions are listed in decreasing order of preference (from top to bottom within the table and from left to right within each table row). Conversions that are not listed are not allowed.

When the ECMAScript parameter is a boolean, conversion is performed as follows:

**Table 25 : Conversion from ECMAScript boolean**

DVB-J Parameter Type	Conversion from ECMAScript boolean
boolean	Direct conversion.
java.lang.Boolean java.lang.Object	Object of type java.lang.Boolean.
java.lang.String	Converted to "true" or "false".
double, float, long, int, short, char, byte	1, if true. 0, if false.

When the ECMAScript parameter is a number, conversion is performed as follows:

**Table 26 : Conversion from ECMAScript number**

DVB-J Parameter Type	Conversion from ECMAScript Number
double	Direct conversion with full precision.
java.lang.Double	java.lang.Double created with full precision.
Float	Rounded to float precision. Values out of float range become +infinity or -infinity.
long, int, short, char, byte	Rounded to appropriate precision. Values out of range causes the <i>DVBException</i> with the error code <i>CONVERSION_TYPE_ERR</i> to be raised. Passing NaN causes the <i>DVBException</i> with the error code <i>CONVERSION_TYPE_ERR</i> to be raised.
java.lang.String	Conversion to a string using the ECMAScript toString function.
boolean	Passing NaN causes the <i>DVBException</i> with the error code <i>CONVERSION_TYPE_ERR</i> to be raised. 0 is false. Non-zero is true.
java.lang.Object	java.lang.Double created with full precision



When the ECMAScript parameter is a string, conversion is performed as follows:

**Table 27 : Conversion from ECMAScript string**

DVB-J Parameter Type	Conversion from ECMAScript string
java.lang.String java.lang.Object	Object of type java.lang.String.
char	If string is a single character, conversion via the ECMAScript charCodeAt() function; otherwise the <a href="#">DVBException</a> with the error code <a href="#">CONVERSION_TYPE_ERR</a> is raised.

When the ECMAScript parameter is undefined, conversion is performed as follows:

**Table 28 : Conversion from ECMAScript undefined**

DVB-J Parameter Type	Conversion from ECMAScript undefined
java.lang.String java.lang.Object	Object of type java.lang.String with a value of "undefined".
boolean	False
long, int, short, byte, char	Zero
double, float	Nan

When the ECMAScript parameter is null, conversion is performed as follows:

**Table 29 : Conversion from ECMAScript null**

DVB-J Parameter Type	Conversion from ECMAScript null
Any object	Null
boolean	False
double, float, long, int, short, byte, char	Zero

When the ECMAScript parameter is JSONArray, JSONObject or JavaClass, conversion is performed as follows:

**Table 30 : Conversion from ECMAScript DVB-J objects**

DVB-J Parameter Type	Conversion from ECMAScript DVB-J Objects
Most specific assignment compatible class.	Unwrap ECMAScript JSONObject to original DVB-J Object. ECMAScript JSONObject is of an incompatible class to the required class the <a href="#">DVBException</a> with the error code <a href="#">CONVERSION_TYPE_ERR</a> shall be raised
java.lang.String	Result of toString().
double, float, long, int, short, char, byte	Result of doubleValue() is taken and converted as a Number to the DVB-J type as above. If no doubleValue() method is defined, causes the <a href="#">DVBException</a> with the error code <a href="#">CONVERSION_TYPE_ERR</a> to be raised.

When the ECMAScript parameter is an ECMAScript Object, conversion is performed as follows:

**Table 31 : Conversion from ECMAScript Object**

DVB-J Parameter Type	Conversion from ECMAScript Object
java.lang.Object (including subclasses) or interface	If ECMAScript object has all public fields and methods of the DVB-J parameter type, and has been generated as specified in <a href="#">Subclassing and Interface Instance Creation</a> , then the generated DVB-J class is used. Otherwise the <a href="#">DVBException</a> with the error code <a href="#">CONVERSION_TYPE_ERR</a> shall be raised

ECMAScript Arrays are passed as parameters only to Java methods accepting an array as its corresponding argument. An ECMAScript array containing a single element type matches a Java array parameter based on the matching preference of the array element types. An ECMAScript array containing different element types is not convertible to and does not match any Java parameter type.

### 8.10.2.10 Subclassing and Interface Instance Creation

Instances of an instantiable Java class can be created by using the "new" operator on the Java class, e.g.

```
new Packages.java.lang.String("astring");
```

The parameters are passed to the constructor using the type conversion rules above. This mechanism is extended to allow the creation of instances of classes that must be subclassed to produce an instantiable class. The SubType object is defined and used with a constructor function to allow ECMAScript to construct an instance of a subclass. The Subtype constructor takes a string naming the Java class to be subclassed as its argument, e.g. "java.awt.ActionListener" and an ECMAScript function which acts as a constructor to initialize any abstract elements. Invoking the Subtype constructor on a final Java class shall cause the [DVBException](#) with the error code [SUBCLASS\\_NOT\\_ALLOWED\\_ERR](#) to be raised. The object returned by the Subtype constructor is a function used to create instances. When this function is used as a constructor, its arguments are passed to the function specified in the Subtype constructor. The function is invoked with the current object being an ECMAScript handle to the JavaObject being instantiated.

Example:

```
function actDone(event) {
  /* My event handling method */
}

function MyListenerConstructor(arg1, arg2) {
  /* if ActionListener had a constructor it could be called here
  * e.g. this.ActionListener(arg2);
  */
  this.ActionPerformed = arg1;
}

MyListenerType = new Subtype("java.awt.ActionListener", MyListenerConstructor);
MyListener = new MyListenerType(actDone, 0);
```

NOTE: This mechanism may be used to create listeners from ECMAScript.

An example of constructing and setting a listener:

```
<script>
  function receiveEvent(x) {
    window.open("http://chat.tv.com/chatLogin?user="+x, _chatFrame);
  }
  function chatConstructor() {
    this.receiveUserPreferenceChangeEvent = receiveEvent;
  }
  chatListenerType = new Subtype("org.dvb.user.UserPreferenceChangeEvent", chatConstructor);
  chatListener = new chatListenerType();

  Packages.org.dvb.user.UserPreferences.addUserPreferenceChangeListener(chatListener);
</script>
```

### 8.10.2.11 Type Conversion (DVB-J to ECMAScript)

Type conversion of return values occurs as follows:

**Table 32 : Conversion from DVB-J types**

DVB-J Return Value Type	Conversion to ECMAScript type
double, float, long, short, int, byte, char	Direct conversion to number
boolean	Direct conversion to boolean
array	JavaArray
java.lang.Class	JavaClass
Object wrapped around ECMAScript Object	Direct conversion (unwrapping)

**Table 32 : Conversion from DVB-J types**

DVB-J Return Value Type	Conversion to ECMAScript type
Other objects (including java.lang.Double, java.lang.Integer and java.lang.String (note 1), java.lang.Exception)	JavaScript (an ECMAScript wrapper around the DVB-J Object providing the same methods and
NOTE 1: The explicit or implicit invocation of the ECMAScript toString() function on the ECMAScript handle to a java.lang.String shall return an external string.	

### 8.10.2.12 Catching DVB-J Exceptions in ECMAScript

DVB-J exceptions are thrown back into ECMAScript when the underlying DVB-J method throws an exception, or an exception is generated as part of the attempted type conversion from ECMAScript to DVB-J (see type conversions above). For example, the following ECMAScript:

```
try {
    Packages.java.lang.Class.forName("someClass");
} catch (e) {
    print("Exception: " + e);
}
```

would print out something like the following, if someClass does not exist.

Exception: java.lang.ClassNotFoundException: someClass.

## 8.11 Document Object Model (DOM)

DOM is a platform and language neutral interface that provides programmatic access to the content, structure and style of documents. Support for language bindings for the DOM for both Java language and ECMAScript is required.

**Table 33 : Supported DOM modules**

DOM Module		Where specified	Required
Package	Feature String		
Level 2 Core	Core	Document Object Model (DOM) Level 2 Core Specification [82]	Yes
	XML		No
Level 2 Views	Views	Document Object Model (DOM) Level 2 Views Specification [86]	Yes
Level 2 Stylesheets	StyleSheets	Document Object Model (DOM) Level 2 Style Specification [85].	No
Level 2 CSS stylesheets	CSS	Document Object Model (DOM) Level 2 Style Specification [85].	No
	CSS2		Yes
Level 2 Events	Events	Document Object Model (DOM) Level 2 Events Specification [84].	Yes
	UIEvents		Yes
	MutationEvents		Yes
DVB-HTML	DVBHTML	See 8.11.4, "DVB-HTML DOM module" on page 135	Yes
DVB Events	DVBEvents	See 8.11.2, "DVB Events DOM module" on page 129	Yes
DVB Key Events	DVBKeyEvents	See 8.11.3, "DVB Key events DOM module" on page 133	Yes
DVB CSS	DVBCSS	See 8.11.8.1, "DVB CSS DOM module" on page 157	Yes
DVB Environment	DVBEnvironment	See 8.11.7, "DVB Environment object module" on page 153	Yes

### 8.11.1 DOM Level 2 Events

#### 8.11.1.1 Fundamental interfaces

DVB-HTML shall support the events module as defined in [Document Object Model \(DOM\) Level 2 Events Specification \[84\]](#) (which includes the following interfaces: `EventTarget`, `EventListener`, `DocumentEvent`, `Event` and `EventException`).

#### 8.11.1.2 Event interfaces

The following event sets from [Document Object Model \(DOM\) Level 2 Events Specification \[84\]](#) shall be supported:

- `UIEvent`
- `MutationEvent`

It is not required to support the following event sets:

- HTML events

NOTE: Equivalents to the Load and Unload events are provided, see 9.4, "Application activity events" on page 200.

- `MouseEvent`

It is recommended that implementations providing support for mouse like devices provide the `MouseEvent` event set.

## 8.11.2 DVB Events DOM module

A DOM application can use the `hasFeature` method of the DOM implementation interface to determine that this module is supported. The feature string for all interfaces listed in this module is "DVBEvents" and the version "2.0".

### 8.11.2.1 Key events

DVB-HTML will provide the W3C event bindings for keyboard and remote control events in a later release of this specification when the W3C Dom level 3 events specification is finalised. Until such time, scripting will have access to key events as specified in an additional DVB DOM module specified in 8.11.3, "[DVB Key events DOM module](#)" on [page 133](#).

in the following DVB-HTML specific manner:

### 8.11.2.2 Lifecycle events

This section provides the mapping of DVB-HTML life cycle events to the DOM event model.

#### 8.11.2.2.1 Interface DVBLifecycleEvent

The DVBLifecycleEvent interface provides specific contextual information to a DVB-HTML application associated with its lifecycle.

All lifecycle events are delivered to the root element of the document in the root frame of an application and follow the normal event delivery rules in [Document Object Model \(DOM\) Level 2 Events Specification \[84\]](#). For clarification, sub-frames which contain the root of separate applications (as described in [8.14.3, "Inter application security" on page 172](#)) also have DVB lifecycle events delivered to the corresponding document root. The different types of such events that can occur are detailed under [8.11.2.2.2, "Event definitions" on page 130](#).

##### 8.11.2.2.1.1 IDL Definition

```
// Introduced in DVB-HTML:
interface DVBLifecycleEvent : Event {
  readonly attribute long detail;
  void initDVBLifecycleEvent(in DOMString typeArg,
    in boolean canBubbleArg,
    in boolean cancelableArg,
    in long detailArg);
};
```

##### 8.11.2.2.1.2 Attributes

**detail:** This attribute specifies additional information about the event the meaning of which depends on the type of the event.

## 8.11.2.2.1.3 Methods

Table 34 : DVB Lifecycle Event Method

Method	Name	Description		
	initDVBLifecycleEvent	This method is used to initialize the value of a lifecycle event created through the DocumentEvent interface. This method may only be called before the lifecycle event has been dispatched via the <code>dispatchEvent</code> method, though it may be called multiple times during that phase if necessary. If called multiple times, the final invocation takes precedence.		
Parameters	Name	Type	Qualifier	Description
	typeArg	DOMString		Specifies the event type. The string is identical to the name of the event (e.g. the AppStarting event is specified with the string "AppStarting").
	canBubbleArg	Boolean		Specifies whether or not the event can bubble.
	cancelableArg			Specifies whether or not the event's default action can be prevented.
	detailArg	Number		Specifies the Event's detail.
Return value	Type	Description		
No Return Value.				
Exceptions				
No Exceptions.				

## 8.11.2.2.2 Event definitions

## 8.11.2.2.2.1 AppStarting

This event is given to the DVB-HTML application whilst in the Active state if and only if it is signalled as PREFETCH. The user agent sends this event when it receives the [dvb.start event](#). After receiving the [dvb.start event](#) and prior to sending this event the user agent begins displaying the DVB-HTML application.

- Bubbles: No
- Cancelable: No
- Context Info: None

## 8.11.2.2.2.2 AppActive

This event is given to the DVB-HTML application on transitioning from the Loading state to the Active state.

- Bubbles: No
- Cancelable: No
- Context Info: [detail](#)

The values of [detail](#) are those defined in table 80, "DVB-HTML application control code values" on page 216 with the following semantics:

- PREFETCH means that the DVB-HTML application will not be presented by the user agent until the user agent has sent an AppStarting event.
- All other values imply that the DVB-HTML application will not receive an AppStarting event and can assume that presentation has begun.

#### 8.11.2.2.2.3 AppPause

This event is given to the DVB-HTML application on transitioning to the Paused state.

- Bubbles: No
- Cancelable: No
- Context Info: None

#### 8.11.2.2.2.4 AppResume

This event is given to the DVB-HTML application on transitioning out of the Paused state to the Active state.

- Bubbles: No
- Cancelable: No
- Context Info: None

#### 8.11.2.2.2.5 AppDestroyed

This event is given to the DVB-HTML application on transitioning to the Destroyed state.

- Bubbles: No
- Cancelable: No
- Context Info: None

#### 8.11.2.2.2.6 AppKilled

This event is given to the DVB-HTML application on transitioning to the Killed state.

- Bubbles: No
- Cancelable: No
- Context Info: None

#### 8.11.2.2.2.7 AppTerminating

This event is given to a DVB-HTML application when an application within a subframe terminates.

- Bubbles: No
- Cancelable: No
- Context Info: [detail](#)

The value of [detail](#) is the name of the frame where the terminating application was hosted:

### 8.11.2.2.3 State transition summary

The following table summarizes the allowed transitions in the lifecycle model (see 9.3.3, "The State Model" on page 196).

State transition	Loading	Active	Paused	Destroyed	Killed
<b>Loading to</b>	no	yes <sup>1</sup>	no	yes <sup>2</sup>	yes <sup>3</sup>
<b>Active to</b>	no	no	yes <sup>4</sup>	yes <sup>5</sup>	yes <sup>6</sup>
<b>Paused to</b>	no	yes <sup>7</sup>	no	yes <sup>8</sup>	yes <sup>9</sup>
<b>Destroyed to</b>	no	no	no	no	yes <sup>10</sup>
<b>Killed to</b>	no	no	no	no	no

The following table describes the events delivered to the application following these transitions.

NOTE: In 8 and 9 no AppResume event be received.

cross reference	transition	event delivered
<b>1</b>	Loading to Active	event AppActive on entry to Active state
<b>2</b>	Loading to Destroyed	event AppDestroyed on entry to Destroyed state
<b>3</b>	Loading to Killed	event AppKilled on entry to Killed state
<b>4</b>	Active to Paused	event AppPause on exit from Active state
<b>5</b>	Active to Destroyed	event AppDestroyed on entry to Destroyed state
<b>6</b>	Active to Killed	event AppKilled on entry to Killed state
<b>7</b>	Paused to Active	event AppResume on entry to Active state
<b>8</b>	Paused to Destroyed	event AppDestroyed on entry to Destroyed state
<b>9</b>	Paused to Killed	event AppKilled on entry to Killed state
<b>10</b>	Destroyed to Killed	event AppKilled on entry to Killed state

### 8.11.2.3 Additional DVB Events

#### 8.11.2.3.1 Trigger events

See 8.7, "Synchronization" on page 92.

#### 8.11.2.3.2 DVBDOMStable event

This event signals the completion of the current document DOM construction. After the receipt of this event the DOM structure of the document will be complete, however the DOM structure in sub-frames or in-line frames may not be complete and resources such as the contents of images may not have finished loading. Prior to the generating the event implementations are free to have an incomplete DOM structure referenced by the document variable.

This event uses the base Event interface to pass contextual information. The following type is allowed:



**DVBDOMStable:** This event occurs when the DOM implementation is fully constructed, but is not required to wait until all of the associated frames in a <frameset>, content in <image> or <object> elements loads. This event is valid for <body> and <frame> elements.

- Bubbles: No
- Cancelable: No
- Context Info: None

### 8.11.2.3.3 DVB-HTML events

These events use the base Event interface to pass contextual information. The following types are allowed:

**load:** The load event occurs when the DOM implementation finishes loading all content within a document, all frames within a <frame>, or an <object> element.

- Bubbles: No
- Cancelable: No
- Context Info: None

**unload:** The unload event occurs when the DOM implementation removes a document from a window or frame. This event is valid for <body> and <frame> elements.

- Bubbles: No
- Cancelable: No
- Context Info: None

## 8.11.3 DVB Key events DOM module

A DOM application may use the `hasFeature( feature, version )` method of the `DOMImplementation` interface with parameter values "DVBKeyEvents" and "2.0" (respectively) to determine whether or not the event module is supported by the implementation. The feature string is also used with the `createEvent` method. It is implementation dependent how a MHP gets the set of supported keys.

Key events generated by a real or virtual keyboard or from a remote control device shall be generated by the user agent and can be registered for by applications through the DOM. Registering for events using the type string "HKeyEvent" or "HRCEvent" can place a listener that shall receive an object that implements the `DVBKeyEvent` interface. All Key events are initially targeted at the element with focus, or the body element if no element has focus

e.g.

```
function myListener(evt) { ... }
node.addEventListener("HRCEvent", myListener)
```

### 8.11.3.1 Interface DVBKeyEvent

The `DVBKeyEvent` interface provides specific contextual information associated with Remote Control and Keyboard Events. The interface is able to describe the representation of a key event as a string, color or symbol (such as a triangle, ">", for "play"). This allows an application to describe a button on an input device correctly for a given platform.

The `keyChar` attribute describes the representation of a key event as a string. All available events shall have a text representation. Other representations may be available

- NOTE: The `DVBKeyEvent` interface is designed to be compatible with the W3C level 3 DOM key events albeit with some redundancy, and in future versions of the specification it is envisaged that objects will also support the W3C defined interface.

### 8.11.3.1.1 IDL Definition

// Introduced in MHP 1.1:

```
interface DVBKeyEvent : UIEvent {
  readonly attribute unsigned long when;
  readonly attribute unsigned long modifiers;
  readonly attribute unsigned long keyCode;
  readonly attribute DOMString keyChar;
  readonly attribute DOMString color;
  readonly attribute DOMString symbol;
  void initDVBKeyEvent(in DOMString type,
    in boolean canBubbleArg,
    in boolean cancelableArg,
    in unsigned long modifiers,
    in unsigned long keycode,
  );
};
```

## 8.11.3.1.2 Attributes

Table 35 : Key event attributes

Name	Readonly	Type	Description
when	yes	unsigned long	The timestamp for the event
modifiers	yes	unsigned long	Indication of any modification keys active for the event
keyCode	yes	unsigned long	The key code of the key associates with this event.
keyChar	yes	unsigned long	The character representation of the key associated with this event
color	yes	DOMString	The six coloured key events (VK_COLORED_KEY_0, ..., VK_COLORED_KEY_5), if implemented, must also be represented by a color. For all other key events this value should be null. The format of the color is specified in <a href="#">CSS 2 [87]</a> section 4.3.6 Colors.
symbol	yes	DOMString	The URL to an image provided by a MHP to describe the symbol associated with a certain key. The recommended symbols for some Key events are described in <a href="#">HAVI [50]</a> . The value of this attribute can be NULL.

## 8.11.3.1.3 Methods

Table 36 : Key Event Method

Method	Name	Description		
	initDVBKeyEvent	This method is used to initialize the value of a keyboard event. This method may only be called before the key event has been dispatched via the <code>dispatchEvent</code> method, though it may be called multiple times during that phase if necessary. If called multiple times, the final invocation takes precedence.		
Parameters	Name	Type	Qualifier	Description
	type	DOMString	-	Either "HRCEvent" or "HKeyEvent"
	canBubbleArg	boolean		Specifies whether or not the event can bubble.
	cancelableArg	boolean		Specifies whether or not the event's default action can be prevented
	modifiers	unsigned long		The modifiers to be applied to the event
	keycode	unsigned long	-	The Keycode of the event to be generated
Return value	Type	Description		
	-	Initialises the event, all of the attributes shall be set as if this event were generated by the system.		

## 8.11.4 DVB-HTML DOM module

## 8.11.4.1 Conformance

A DVB-HTML user agent shall be considered as being an HTML-only DOM implementation as defined in the [W3C Document Object Model \(DOM\) Level 2 Core Specification \[82\]](#), so that conformance statements in DOM Level 2

applying to those kinds of implementation will apply also to a DVB-HTML user agent. E.g. a DVB-HTML DOM implementation is not required to implement the `createDocument()` method in the `DOMImplementation` interface from the DOM Level 2 Core.

#### 8.11.4.2 Differences from W3C DOM Level 1 HTML interfaces

The W3C DOM 1 HTML interfaces were designed in order to provide convenience functions to manipulate HTML 4.0 documents and also to create documents from scratch. The DOM DVB-HTML module has been defined with the objective to manipulate DVB-HTML documents without any intention to create them. In general, the DOM DVB-HTML module follows the design of the W3C DOM1 HTML interfaces, however the following differences can be observed:

- Some attributes have been made immutable in the DOM DVB-HTML module. These modifications have been indicated in this document.
- Some attributes and interfaces have not been included in the DOM DVB-HTML module.
- For the length attribute in all interfaces, the type has been homogenised to unsigned long.
- Some semantics have been added to clarify user agent behaviour (e.g. on modification of the disabled attribute)
- Some semantics have been altered to take into account the DVB-HTML DTD (e.g. since the name attribute is not supported for identification purposes some attribute semantics had to be modified accordingly).

#### 8.11.4.3 Extensions

##### 8.11.4.3.1 Enumerations

In DVB-HTML, some element attributes can be set to a limited number of values. In order to be as much as possible in line with the IDL defined in the W3C DOM 1 HTML, this specification does not use the IDL `enum`. For attributes that take a limited set of values (i.e. an enumeration) the following semantics shall apply:

- If a mutable attribute is set with a value outside of the possible values the `DOMException` with the error code `SYNTAX_ERR` is raised and the original value shall be kept.

The bindings reflect this choice, see annex AC, "(normative): ECMAScript Binding" on page 1283 and annex AD, "(normative): Support for DVB-HTML" on page 1296.

##### 8.11.4.3.2 Initial and current values of form controls

Form controls that accept a user input, e.g. text area, option, text, password, checkbox, radio button, have both an initial value and a current value. The names and the types of these values depend on the type of the control. When a control is created, the means by which the initial value is set depends on the control. See the corresponding interface definition 8.11.4.6.6, "DVBHTMLFormElement Interface" on page 143.

When a control is created, the current value is set to the initial value.

The initial value can only be modified by a script. The current value can be modified either by a user interaction or a scripting instruction. If the current value is modified, the display shall be updated to reflect the value changed.

When a form is reset, either by user interaction or by scripting instruction, for each of the controls accepting user input and contained in the form, the current value shall be set to the initial value. The initial value may have been modified between the creation of the control and the reset of the form. As such, the initial value used shall be the value at the time of the reset.

Form controls that do not accept user input, e.g. hidden controls, submit buttons, reset buttons, buttons, submit images, have only one value, a current value. When a control is created, the means by which the current value is set depends on the control. See 8.11.4.6.6, "DVBHTMLFormElement Interface" on page 143. The current value can only be modified by a scripting instruction. In such a case, it is not required to affect the display.

The file input control has a different behaviour from those described above. It accepts a user input but only has a current value. The semantics of the current value of form controls that do not accept user input applies.

#### 8.11.4.4 System aspects

##### 8.11.4.4.1 Access to the document

###### 8.11.4.4.1.1 Java

The DVB DOM Implementation shall implement the `org.dvb.dom` API as defined in 11.13.1, "Document object model APIs" on page 282. This will provide applications a handle to the DVB-HTML document object model.

###### 8.11.4.4.1.2 ECMAScript

The DVB DOM implementation shall provide to applications a `document` object that represents the current DVB-HTML document.

##### 8.11.4.4.2 DOM DVB-HTML module

A DOM application can use the `hasFeature` method of the `DOMImplementation` interface to determine whether this module is supported or not. The feature string for all the interfaces listed in this module is "DVBHTML" and the version is "2.0".

##### 8.11.4.4.3 DOM modification

###### 8.11.4.4.3.1 DOM access

Any DOM call from the DOM DVB-HTML module shall be synchronous. As such, any subsequent calls can assume that they can operate on a DOM structure updated accordingly to the previous call.

###### 8.11.4.4.3.2 Synchronisation with renderer actions.

Wherever applicable, any required impact on the visual rendering following a DOM call has been described. However, this specification is voluntarily silent on the nature of the relationships between a DOM modification and the renderer, in particular on the synchronisation aspects.

###### 8.11.4.4.3.3 Modifications from the DOM Core

A `NO_MODIFICATION_ALLOWED_ERR` exception shall be raised when attempting to modify an attribute from the DOM Core which equivalent attribute in the DOM DVB-HTML module is defined to be `readonly`, e.g. this exception shall be raised when modifying the `type` attribute in the `button` element through the DOM Core.

#### 8.11.4.5 Miscellaneous interfaces

This section describes various utility interfaces of the DOM DVB-HTML module.

##### 8.11.4.5.1 DVB-HTMLCollection Interface

A `DVBHTMLCollection` is a list of DVB-HTML elements. An individual element may be accessed by either ordinal index or the element's `id` attribute.

NOTE: Collections in the DVB-HTML DOM are assumed to be dynamic meaning that they are automatically updated when the underlying document is modified.

###### 8.11.4.5.1.1 IDL Definition

```
interface DVBHTMLCollection {
    readonly attribute unsigned long length;
    Node namedItem(in DOMString name);
    Node item(in unsigned long index);
};
```

The semantics of the attributes and methods associated with this interface are the same as the semantics associated with the `HTMLCollection` interface defined in [Document Object Model \(DOM\) Level 1 Specification \[83\]](#).

### 8.11.4.5.2 DVBHTMLDocument Interface

A `DVBHTMLDocument` is the root of the DVB-HTML document hierarchy and holds the entire content. Besides providing access to the hierarchy, it also provides some convenience methods for accessing certain sets of information from the document.

#### 8.11.4.5.2.1 IDL Definition

```
interface DVBHTMLDocument : Document {
    readonly attribute DVBHTMLCollection anchors; // DVB-HTML Type coherence
    attribute DVBHTMLCollection body; // DVB-HTML Type coherence
    attribute DOMString cookie;
    readonly attribute DOMString domain;
    readonly attribute DVBHTMLCollection forms; // DVB-HTML Type coherence
    readonly attribute DVBHTMLCollection images; // DVB-HTML Type coherence
    readonly attribute DVBHTMLCollection links; // DVB-HTML Type coherence
    readonly attribute DOMString referrer;
    attribute DOMString title;
    readonly attribute DOMString URL;

    void open();
    void close();
    void write(in DOMString text);
    void writeln(in DOMString text);
};
```

The semantics of the attributes and methods associated with this interface are the same as the semantics associated with the `HTMLDocument` interface defined in [Document Object Model \(DOM\) Level 1 Specification \[83\]](#), with the extensions, restrictions or amendments defined in the following subsections.

## 8.11.4.5.2.2

## Attributes

**Table 37 : DVBHTMLDocument attributes**

Name	Readonly	Type	Description
anchors	yes	DVBHTMLCollection	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Only the return type differs
body	no	DVBHTMLCollection	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Only the return type differs
domain	yes	DOMString	See domain in 8.16.6, "Domain" on page 182.
forms	yes	DVBHTMLCollection	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Only the return type differs
images	yes	DVBHTMLCollection	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Only the return type differs
links	yes	DVBHTMLCollection	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Only the return type differs
title	no	DOMString	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . In addition, if the user agent makes this visible, then it shall update the rendering upon modification. This attribute is linked to the <title> element changes to one shall be reflected in the other.

## 8.11.4.5.2.3

## Methods

**Table 38 : DVBHTMLDocument open method**

Method	Name	Description		
	open()	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> opens the document for writing. This creates a new empty document in the target to which subsequent writes are made. The transition and the rendering of the new document shall not be performed until a call to the close() method. The ECMAScript context which opened the new document cannot be destroyed until it calls document.close(). (note 1)		
Parameters	Name	Type	Qualifier	Description
Return value	Type	Description		
	-	This method does not return a value.		
NOTE 1: document.open() shall not change the URL of the document.				

**Table 39 : DVBHTMLDocument close method**

Method	Name	Description		
	close()	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Rendering of the document is performed as if a link had been followed with that document as a target.		
Parameters	Name	Type	Qualifier	Description
Return value	Type	Description		
	-	This method does not return a value.		

Table 40 : DVBHTMLDocument write method

Method	Name	Description		
	write()	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . An inline non-deferred script performing document.write() feeds additional strings to the parser. The text is parsed into the document's structure model. These strings are sent to the parser sequenced immediately after the closing script tag of the inline script. See also (, "Implicit document.open()" on page 140).		
Parameters	Name	Type	Qualifier	Description
Return value	Type	Description		
	-	This method does not return a value.		

Table 41 : DVBHTMLDocument writeln method

Method	Name	Description		
	writeln()	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . An inline non-deferred script performing document.writeln() feeds additional strings to the parser. The text is parsed into the document's structure model. These strings are sent to the parser sequenced immediately after the closing script tag of the inline script. See also (, "Implicit document.open()" on page 140).		
Parameters	Name	Type	Qualifier	Description
Return value	Type	Description		
	-	This method does not return a value.		

**Implicit document.open()**

If document.write() is called from a deferred script context and the document has not been explicitly opened for writing, the first write() performs an implicit open of the containing document.

**Implicit document.close()**

If the deferred inline script context which performed an open (explicit or implicit) exits without calling close on the document, the document is closed implicitly.

When a document is trying to access the value of the forms, anchors, links or images attributes before the parsing of the document is complete, the returned value shall be the collection of matching elements at the time of the call.

**8.11.4.6 DVB-HTML element related interfaces****8.11.4.6.1 DVBHTMLElement Interface**

All DVB-HTML element interfaces derive from this class.

**8.11.4.6.1.1 IDL Definition**

```
interface DVBHTMLElement : Element {
    attribute DOMString className;
    attribute DOMString dir;
    attribute DOMString id;
    attribute DOMString lang;
    attribute DOMString title;
};
```



### 8.11.4.6.1.2 Attributes

The semantics of the attributes associated with this interface are the same as the semantics associated with the `HTMLInputElement` interface defined in [Document Object Model \(DOM\) Level 1 Specification \[83\]](#), with the extensions, restrictions or amendments defined in the following subsection.

**Table 42 :**

Name	ReadOnly	Type	Description
dir	no	DOMString	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Trying to set this attribute to a value other than the values allowed in <a href="#">HTML 4 [90]</a> shall cause the <code>DOMException</code> with the error code <code>SYNTAX_ERR</code> to be raised.
lang	no	DOMString	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Trying to set this attribute to a value other than the values allowed in <a href="#">HTML 4 [90]</a> shall cause the <code>DOMException</code> with the error code <code>SYNTAX_ERR</code> to be raised.

### 8.11.4.6.2 DVBHTMLAnchorElement Interface

This interface models the anchor element, corresponding to the DVB-HTML `<a>` tag. It represents a source anchor, the `<A>` tag with the `href` attribute, or a destination anchor, the `<a>` tag with the `id` attribute.

#### 8.11.4.6.2.1 IDL Definition

```
interface DVBHTMLAnchorElement : DVBHTMLInputElement {
    attribute DOMString accessKey;
    attribute DOMString charset;
    attribute DOMString href;
    attribute DOMString hreflang;
    attribute DOMString target; // Diverge from W3C DOM HTML
    attribute DOMString type;

    void blur();
    void focus();
};
```

The semantics of the attributes and methods associated with this interface are the same as the semantics associated with the `HTMLAnchorElement` interface defined in [Document Object Model \(DOM\) Level 1 Specification \[83\]](#), with the extensions, restrictions or amendments defined in the following subsections.

#### 8.11.4.6.2.2 Attributes

**Table 43 :**

Name	ReadOnly	Type	Description
target	no	DOMString	The id of the frame to render the resource designated by the URI in. The semantics of the <code>target</code> attribute in <a href="#">HTML 4 [90]</a> apply with the following modification: all references to the <code>name</code> attribute shall be seen as being references to the <code>id</code> attribute.

### 8.11.4.6.3 DVBHTMLMapElement Interface

This interface models a client-side image map. See the MAP element definition in [HTML 4 \[90\]](#).

#### 8.11.4.6.3.1 IDL definition

```
interface DVBHTMLMapElement : DVBHTMLInputElement {
    readonly attribute DVBHTMLCollection areas; // DVB-HTML Type coherence
};
```

The semantics of the attributes associated with this interface are the same as the semantics associated with the `HTMLMapElement` interface defined in [Document Object Model \(DOM\) Level 1 Specification \[83\]](#), with the extensions, restrictions or amendments defined in the following subsections.

#### 8.11.4.6.3.2 Attributes

**Table 44 :**

Name	ReadOnly	Type	Description
areas	yes	DVBHTMLCollection	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Its type has been changed to DVBHTMLCollection.

NOTE: The name attribute in the map element is deprecated in DVB-HTML. As such, the map is identified only through the `id` attribute.

When a document is trying to access the value of the `areas` attribute before the parsing of the contents of the map element is complete, the returned value shall be the collection of matching elements at the time of the call.

#### 8.11.4.6.4 DVBHTMLAreaElement Interface

This interface models an image map area which can triggers an action when the user clicks on it.

##### 8.11.4.6.4.1 IDL Definition

```
interface DVBHTMLAreaElement : DVBHTMLElement {
    attribute DOMString accessKey;
    attribute DOMString alt;
    attribute DOMString href;
    attribute boolean nohref;
    attribute DOMString target; // Diverge from W3C DOM HTML
};
```

The semantics of the attributes associated with this interface are the same as the semantics associated with the `HTMLAreaElement` interface defined in [Document Object Model \(DOM\) Level 1 Specification \[83\]](#), with the extensions, restrictions or amendments defined in the following subsections.

#### 8.11.4.6.4.2 Attributes

**Table 45 :**

Name	ReadOnly	Type	Description
target	no	DOMString	The id of the frame to render the resource designated by the URI in. The semantics of the target attribute in <a href="#">HTML 4 [90]</a> apply with the following modification : all references to the name attribute shall be seen as being references to the id attribute.

#### 8.11.4.6.5 DVBHTMLButtonElement Interface

`DVBHTMLButtonElement` represents the DVB-HTML `<button>` tag.

##### 8.11.4.6.5.1 IDL Definition

```
interface DVBHTMLButtonElement : DVBHTMLElement {
    attribute DOMString accessKey;
    attribute boolean disabled;
    readonly attribute DVBHTMLFormElement form; // DVB-HTML Type coherence
    attribute DOMString name;
    readonly attribute DOMString type;
    attribute DOMString value;

    void blur(); // Introduced in DVB-HTML DOM
    void focus(); // Introduced in DVB-HTML DOM
};
```

The semantics of the attributes and methods associated with this interface are the same as the semantics associated with the `HTMLButtonElement` interface defined in [Document Object Model \(DOM\) Level 1 Specification \[83\]](#), with the extensions, restrictions or amendments defined in the following subsections.

#### 8.11.4.6.5.2 Attributes

**Table 46 :**

Name	Readonly	Type	Description
form	yes	DVBHTMLFormElement	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Only the return type differs.
value	no	DOMString	The button's current value. It corresponds to the value attribute of the DVB-HTML <code>&lt;button&gt;</code> tag but shall be interpreted as a current value (as defined in section 8.11.4.3.2, "Initial and current values of form controls" on page 136).

#### 8.11.4.6.5.3 Methods

**Table 47 :**

Method	Name	Description		
	blur	Removes focus from this element.		
Parameters	Name	Type	Qualifier	Description
	-	-	-	-
Return value	Type	Description		
	-	-		

**Table 48 :**

Method	Name	Description		
	focus	Gives focus to this element.		
Parameters	Name	Type	Qualifier	Description
	-	-	-	-
Return value	Type	Description		
	-	-		

#### 8.11.4.6.6 DVBHTMLFormElement Interface

The `DVBHTMLFormElement` encompasses behaviour similar to a collection and an element. It provides direct access to the contained input elements as well as the attributes of the form element. See the `FormElement` definition in [HTML 4 \[90\]](#).

##### 8.11.4.6.6.1 IDL Definition

```
interface DVBHTMLFormElement : DVBHTMLFormElement {
    attribute DOMString action;
    attribute DOMString acceptCharset;
    attribute DOMString enctype;
    readonly attribute DVBHTMLCollection elements; // DVB-HTML Type coherence
    readonly attribute unsigned long length; // Lengths type coherence
    attribute DOMString method;
    attribute DOMString target; // Diverge from W3C DOM HTML

    void reset();
    void submit();
}
```

The semantics of the attributes and methods associated with this interface are the same as the semantics associated with the `HTMLFormElement` interface defined in [Document Object Model \(DOM\) Level 1 Specification \[83\]](#), with the extensions, restrictions or amendments defined in the following subsections.

#### 8.11.4.6.6.2 Attributes

**Table 49**

Name	Readonly	Type	Description
elements	yes	DVBHTMLCollection	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Only the type differs
length	yes	unsigned long	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Only the type differs
method	no	DOMString	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Trying to set this attribute to a value other than the values allowed in <a href="#">HTML 4 [90]</a> shall cause the <code>DOMException</code> with the error code <code>SYNTAX_ERR</code> to be raised.
target	no	DOMString	The id of the frame to render the resource designated by the URI in. The semantics of the target attribute in HTML 4.01 apply with the following modification: all references to the name attribute shall be seen as being references to the id attribute.

When a document is trying to access the value of the `elements` attribute before the parsing of the contents of the form element is complete, the returned value shall be the collection of matching elements at the time of the call.

#### 8.11.4.6.7 DVBHTMLFrameElement Interface

##### 8.11.4.6.7.1 IDL Definition

```
interface DVBHTMLFrameElement : DVBHTMLElement {
    readonly attribute Document contentDocument;
    readonly attribute DOMString frameBorder; // Diverge from W3C DOM HTML
    attribute DOMString longDesc;
    readonly attribute DOMString marginHeight // Diverge from W3C DOM HTML
    readonly attribute DOMString marginWidth; // Diverge from W3C DOM HTML
    readonly attribute DOMString scrolling; // Diverge from W3C DOM HTML
    attribute DOMString src;
};
```

The semantics of the attributes associated with this interface are the same as the semantics associated with the `HTMLFrameElement` interface defined in [Document Object Model \(DOM\) Level 1 Specification \[83\]](#), with the extensions, restrictions or amendments defined in the following subsections.

#### 8.11.4.6.7.2 Attributes

**Table 50**

Name	Readonly	Type	Description
frameBorder	yes	DOMString	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . This attribute has been moved to readonly.
marginHeight	yes	DOMString	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . This attribute has been moved to readonly.
marginWidth	yes	DOMString	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . This attribute has been moved to readonly.
scrolling	yes	DOMString	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . This attribute has been moved to readonly.

### 8.11.4.6.8 DVBHTMLFrameSetElement Interface

#### 8.11.4.6.8.1 IDL Definition

```
interface DVBHTMLFrameSetElement : DVBHTMLMElement {
    readonly attribute DOMString cols; // Diverge from W3C DOM HTML
    readonly attribute DOMString rows; // Diverge from W3C DOM HTML
};
```

The semantics of the attributes associated with this interface are the same as the semantics associated with the `HTMLFrameSetElement` interface defined in [Document Object Model \(DOM\) Level 1 Specification \[83\]](#), with the extensions, restrictions or amendments defined in the following subsections.

#### 8.11.4.6.8.2 Attributes

**Table 51**

Name	Readonly	Type	Description
cols	yes	DOMString	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . This attribute has been moved to readonly.
rows	yes	DOMString	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . This attribute has been moved to readonly.

### 8.11.4.6.9 DVBHTMLIFrameElement Interface

#### 8.11.4.6.9.1 IDL Definition

```
interface DVBHTMLIFrameElement : DVBHTMLMElement {
    attribute DOMString frameBorder;
    attribute DOMString longDesc;
    attribute DOMString scrolling;
    attribute DOMString src;
    attribute DVBHTMLDocument contentDocument;
};
```

The semantics of the attributes associated with this interface are the same as the semantics associated with the `HTMLIFrameElement` interface defined in [Document Object Model \(DOM\) Level 1 Specification \[83\]](#) Attributes

**Table 52 : IFrame element attributes**

Name	Readonly	Type	Description
contentDocument	no	DVBHTMLDocument	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Only the type differs

### 8.11.4.6.10 DVBHTMLImageElement Interface

#### 8.11.4.6.10.1 IDL Definition

```
interface DVBHTMLImageElement : DVBHTMLMElement {
    attribute DOMString alt;
    attribute DOMString height; // Extension to W3C DOM HTML
    attribute DOMString longDesc;
    attribute DOMString lowSrc;
    attribute DOMString src;
    readonly attribute DOMString useMap; // Diverge from W3C DOM HTML
    attribute DOMString width;
};
```

The semantics of the attributes associated with this interface are the same as the semantics associated with the `HTMLImageElement` interface defined in [Document Object Model \(DOM\) Level 1 Specification \[83\]](#), with the extensions, restrictions or amendments defined in the following subsections.

## 8.11.4.6.10.2 Attributes

Table 53

Name	Readonly	Type	Description
longdesc	no	DOMString	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> , A DVB-HTML User Agent may optionally make use of this resource in device specific ways (e.g for accessibility). The externally referenced resource will not form part of the document model.
useMap	yes	DOMString	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . This attribute has been moved to readonly. The attribute value shall be the id value and not the name value of the associated map element (since the name attribute in the map element is not supported in DVB-HTML)

## 8.11.4.6.11 DVBHTMLObjectElement Interface

## 8.11.4.6.11.1 IDL Definition

```
interface DVBHTMLObjectElement : DVBHTMLElement {
  attribute DOMString archive;
  attribute DOMString code;
  attribute DOMString codeBase;
  attribute DOMString codeType;
  readonly attribute DVBHTMLDocument contentDocument; // Introduced in DVB-HTML:
  attribute DOMString data;
  attribute boolean declare;
  readonly attribute DVBHTMLFormElement form;
  attribute DOMString height;
  attribute DOMString name;
  attribute DOMString standby;
  attribute DOMString type;
  readonly attribute DOMString useMap;
  attribute DOMString width;
};
```

The semantics of the attributes associated with this interface are the same as the semantics associated with the HTMLObjectElement interface defined in [Document Object Model \(DOM\) Level 1 Specification \[83\]](#), with the addition of the contentDocument attribute defined in the following subsection.

## 8.11.4.6.11.2 Attributes

Table 54

Name	Readonly	Type	Description
contentDocument	yes	DOMString	The document this object contains, if there is any and it is available, or null otherwise.

## 8.11.4.6.12 DVBHTMLInputElement Interface

### 8.11.4.6.12.1 IDL Definition

```
interface DVBHTMLInputElement : DVBHTMLInputElement {  
  
    attribute DOMString accept;  
    attribute DOMString accessKey;  
    attribute DOMString alt;  
    attribute boolean checked; // Extension to W3C DOM HTML  
    attribute DOMString defaultValue; // Extension to W3C DOM HTML  
    attribute boolean defaultChecked; // Extension to W3C DOM HTML  
    attribute boolean disabled; // Extension to W3C DOM HTML  
    readonly attribute DVBHTMLFormElement form; // DVB-HTML Type coherence  
    attribute long maxLength;  
    attribute DOMString name;  
    attribute boolean readOnly; // Extension to W3C DOM HTML  
    attribute DOMString size;  
    attribute DOMString src;  
    readonly attribute DOMString type; // Diverge from W3C DOM HTML  
    readonly attribute DOMString useMap; // Diverge from W3C DOM HTML  
    attribute DOMString value; // Extension to W3C DOM HTML  
  
    void blur();  
    void click();  
    void focus();  
    void select();  
};
```

The semantics of the attributes and methods associated with this interface are the same as the semantics associated with the `HTMLInputElement` interface defined in [Document Object Model \(DOM\) Level 1 Specification \[83\]](#), with the extensions, restrictions or amendments defined in the following subsections.

## 8.11.4.6.12.2

## Attributes

Table 55 : (Sheet 1 of 2)

Name	Readonly	Type	Description
checked	no	boolean	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> ; this attribute corresponds to the current value of controls (as defined in section 8.11.4.3.2, "Initial and current values of form controls" on page 136) with attribute type equals to checkbox or radio. It shall be ignored for the other types of control.
defaultChecked	no	boolean	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> ; this attribute corresponds to the initial value of controls (as defined in section 8.11.4.3.2, "Initial and current values of form controls" on page 136) with attribute type equals to checkbox or radio. It is initialised by the checked attribute of the <input> tag. It shall be ignored for the other types of control.
defaultValue	no	DOMString	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> ; this attribute corresponds to the initial value of controls (as defined in section 8.11.4.3.2, "Initial and current values of form controls" on page 136) with attribute type equals to text or password. It is initialised by the value attribute of the <input> element.
disabled	no	boolean	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . If set to true, the effects are the following : Input does not receive the focus. Input is not included in the tabbing navigation. Input is not successful. Since the readOnly state is less restrictive than the disabled state, if the disabled attribute is set to true, the control evolves to the disabled state, independently of the readOnly attribute value. If the disabled attribute is set to false, the control will evolve to the readOnly state depending on the value of the readOnly attribute. By default, the attribute value is FALSE.
form	yes	DVBHTMLFormElement	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> , only the return type differs.
readOnly	no	boolean	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . When set to true and textarea is not in the disabled state, the effects are the following : Input receives the focus but cannot be modified by the user. Input is included in the tabbing navigation. Input is successful. Since the readOnly state is less restrictive than the disabled state, if the readOnly attribute is set to true when the disabled attribute is equal to true, the control remains in the disabled state. By default, the attribute value is FALSE.
type	yes	DOMString	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . This attribute has been moved to readonly.



Table 55 : (Sheet 2 of 2)

Name	Readonly	Type	Description
useMap	yes	DOMString	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . This attribute has been moved to readonly. The attribute value shall be the id value and not the name value of the associated map element (since the name attribute in the map element is not supported in DVB-HTML)
value	no	DOMString	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . This attribute corresponds to the current value of controls (as defined in section 8.11.4.3.2, "Initial and current values of form controls" on page 136 with attribute type equals to text, password, hidden, submit, reset, button, image, or file. For all the controls, it represents the value of the name/value pair sent to the server when the form containing this control is submitted. This attribute is initialised by the value attribute of the <input> tag.

#### 8.11.4.6.13 DVBHTMLOptionElement Interface

This interface models the <option> element.

##### 8.11.4.6.13.1 IDL Definition

```
interface DVBHTMLOptionElement : DVBHTMLInputElement {
    attribute boolean defaultSelected; // Extension to W3C DOM HTML
    attribute boolean disabled;
    readonly attribute DVBHTMLFormElement form; // DVB-HTML Type coherence
    readonly attribute long index;
    attribute DOMString label;
    attribute boolean selected; // Extension to W3C DOM HTML
    readonly attribute DOMString text;
    attribute DOMString value;
};
```

The semantics of the attributes associated with this interface are the same as the semantics associated with the HTMLOptionElement interface defined in [Document Object Model \(DOM\) Level 1 Specification \[83\]](#), with the extensions, restrictions or amendments defined in the following subsections.

## 8.11.4.6.13.2 Attributes

Table 56

Name	ReadOnly	Type	Description
defaultSelected	no	boolean	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . The initial value is as defined in section 8.11.4.3.2, "Initial and current values of form controls" on page 136. If the option is included in a single choice select, it is initialised to true if the attribute checked is present in the option tag and if this option is the latest option between those contained in the select with the attribute checked set. Its value is false otherwise. If the option is included in a multiple choice select, it is initialised to true if the attribute checked is present in the option tag, false otherwise.
form	yes	DVBHTMLFormElement	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Only the return type differs
selected	yes	boolean	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . The current value is as defined in section 8.11.4.3.2, "Initial and current values of form controls" on page 136.

## 8.11.4.6.14 DVBHTMLSelectElement Interface

This interface represents the <select> control which allows the selection of one or multiple options among a list. Some graphical user agents represent single and multiple selection with different types of widgets. Options can be grouped inside an option group, for presentational reasons, or directly inserted in the selection. Option groups can't be nested.

## 8.11.4.6.14.1 IDL Definition

```
interface DVBHTMLSelectElement : DVBHTMLFormElement {
    attribute boolean disabled;
    readonly attribute DVBHTMLFormElement form; // DVB-HTML Type coherence
    readonly attribute unsigned long length; // Lengths type coherence
    readonly attribute boolean multiple; // Diverge from W3C DOM HTML
    attribute DOMString name;
    readonly attribute DVBHTMLCollection options; // DVB-HTML Type coherence
    attribute long selectedIndex;
        // Extension to W3C DOM HTML
    attribute unsigned long size; // Size type coherence
    readonly attribute DOMString type;
    attribute DOMString value;

    void add(in DVBHTMLFormElement element, in DVBHTMLFormElement before) raises(DOMException);
    void blur();
    void focus();
    void remove(in long index);
};
```

The semantics of the attributes and methods associated with this interface are the same as the semantics associated with the `HTMLSelectElement` interface defined in [Document Object Model \(DOM\) Level 1 Specification \[83\]](#), with the extensions, restrictions or amendments defined in the following subsections.

## 8.11.4.6.14.2 Attributes

Table 57

Name	ReadOnly	Type	Description
form	yes	DVBHTMLFormElement	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Only the return type differs
length	yes	unsigned long	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Only the return type differs
multiple	yes	boolean	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . This attribute has been moved to readonly.
options	yes	DVBHTMLCollection	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Only the return type differs.
selectedIndex	no	long	The ordinal index of the selected option, starting from 0. The value -1 is returned if no element is selected. If multiple options are selected, the index of the first selected option is returned. Upon modification, the display shall be affected.
size	no	unsigned long	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Only the return type differs.

When a document is trying to access the value of the `options` attribute before the parsing of the contents of the `select` element is complete, the returned value shall be the collection of matching elements at the time of the call.

## 8.11.4.6.15 DVBHTMLTextAreaElement Interface

This interface represents a multiline text area. It refers to the `<textarea>` DVB-HTML tag.

## 8.11.4.6.15.1 IDL Definition

```
interface DVBHTMLTextArea : DVBHTMLFormElement {
    attribute DOMString accessKey;
    readonly attribute unsigned long cols; // Diverge from W3C DOM HTML
    attribute DOMString defaultValue;
        // Extension to W3C DOM HTML
    attribute boolean disabled; // Extension to W3C DOM HTML
    readonly attribute DVBHTMLFormElement form; // DVB-HTML Type coherence
    attribute DOMString name;
    attribute boolean readOnly // Extension to W3C DOM HTML
    readonly attribute unsigned long rows; // Diverge from W3C DOM HTML
    readonly attribute DOMString type;
    attribute DOMString value; // Extension to W3C DOM HTML

    void blur();
    void focus();
    void select();
}
```

The semantics of the attributes and methods associated with this interface are the same as the semantics associated with the `HTMLTextAreaElement` interface defined in [Document Object Model \(DOM\) Level 1 Specification \[83\]](#), with the extensions, restrictions or amendments defined in the following subsections.

## 8.11.4.6.15.2

## Attributes

Table 58

Name	ReadOnly	Type	Description
cols	yes	unsigned long	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Only the return type differs and it has been moved to readonly.
defaultValue	no	DOMString	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . The initial value (as defined in section 8.11.4.3.2, "Initial and current values of form controls" on page 136) is initialised by the text inserted between the start and end tags of the element.
disabled	no	boolean	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . If set to true, the effects are the following : <ul style="list-style-type: none"> <li>- Textarea does not receive the focus.</li> <li>- Textarea is not included in the tabbing navigation.</li> <li>- Textarea is not successful.</li> </ul> Since the readOnly state is less restrictive than the disabled state, if the disabled attribute is set to true, the control evolves to the disabled state, independently of the readOnly attribute value. If the disabled attribute is set to false, the control will evolve to the readOnly state depending on the value of the readOnly attribute. By default, the attribute value is FALSE.
form	yes	DVBHTMLFormElement	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Only the return type differs.
readOnly	no	boolean	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . If set to true and textarea is not in the disabled state, the effects are the following : <ul style="list-style-type: none"> <li>- Textarea receives the focus but cannot be modified by the user.</li> <li>- Textarea is included in the tabbing navigation.</li> <li>- Textarea is successful.</li> </ul> Since the readOnly state is less restrictive than the disabled state, if the readOnly attribute is set to true when the disabled attribute is equal to true, the control remains in the disabled state. By default, the attribute value is FALSE.
rows	yes	unsigned long	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . Only the return type differs and it has been moved to readonly.
value	no	DOMString	As defined in <a href="#">Document Object Model (DOM) Level 1 Specification [83]</a> . The current value is as defined in section 8.11.4.3.2, "Initial and current values of form controls" on page 136.

## 8.11.5 DVB Exceptions

DOM operations may raise exceptions in "exceptional" circumstances. In general, DVB-HTML will raise exceptions defined by [Document Object Model \(DOM\) Level 2 Core Specification \[82\]](#), however certain methods in DVB-HTML raise other exceptions under other circumstances.

In DVB-HTML the `DVBException` is defined, this is in addition to the `DOMException` of DOM 2 and defines additional error codes.

NOTE: The error codes defined for `DVBException` are defined so as not to clash with those defined for `DOMException`, however the W3C retains the right to define additional codes for `DOMException` which may conflict with `DVBException` in the future. To avoid conflicts the programmer may use typeof: the `DVBException`, as a host object, has a defined type different to that of `DOMException`.

### 8.11.5.1 DVBException

The value returned by the **typeof** operator for DVBException shall be "**dvbexception**".

#### 8.11.5.1.1 IDL Definition

```
exception DVBException {
  unsigned short code;
};

// ExceptionCode
// Introduced in DVB-HTML:
const unsigned short NON_CONFORMANT_ERR = 601;
const unsigned short SECURITY_VIOLATION_ERR = 602;
const unsigned short WINDOW_NOT_FOUND_ERR = 603;
const unsigned short SUBCLASS_NOT_ALLOWED_ERR = 604;
const unsigned short CONVERSION_TYPE_ERR = 605;
```

**ExceptionCode:** An integer indicating the type of error generated.

Other numeric codes are reserved by DVB for possible future use.

#### 8.11.5.1.2 Defined Constants

**NON\_CONFORMANT\_ERR:** An operation attempted to alter the DOM in such a way as would make the document non conformant.

**SECURITY\_VIOLATION\_ERR:** An operation failed because the application did not have sufficient permission to perform it.

**WINDOW\_NOT\_FOUND\_ERR:** The application attempted to open a window which was not part of the application.

**SUBCLASS\_NOT\_ALLOWED\_ERR:** The application attempted to subclass a final class.

**CONVERSION\_TYPE\_ERR:** The application attempted to subclass a final class.

## 8.11.6 Language bindings

### 8.11.6.1 ECMAScript Binding

See annex AC, "(normative): ECMAScript Binding" on page 1283.

### 8.11.6.2 Java Binding

See annex AD, "(normative): Support for DVB-HTML" on page 1296.

## 8.11.7 DVB Environment object module

A DOM application can use the hasFeature method of the DOM implementation interface to determine that this module is supported. The feature string for all interfaces listed in this module is "DVBEnvironment" and the version "2.0".

### 8.11.7.1 Free variables

DVB-HTML shall support ECMAScript access to certain APIs is through a number of variables, which get bound in each page context:

- document
- navigator
- window

Each DVB-HTML application shall see different Navigator, Document and Window objects, Only one Navigator object is created per application lifetime (therefore properties may be set on the Navigator object which persist for the lifetime of the application). The navigator variable is bound to this object in each page context.

Windows and Document are initialized on an as needed basis, Document is valid during and after parsing of the document and is bound to the variable document.

Each window is associated with a single document, there may be more than one extant window object per DVB-HTML application, however multiple window support is included only for frames. There shall be at most one top level window per DVB-HTML application (see [CSS 2 \[87\]](#)).

The window variable is initialized to the immediately containing window for a document.

## 8.11.7.2 Environmental host objects

### 8.11.7.2.1 Navigator Object

Represents the browser itself, the navigator free variable is of this type. This is supported in DVB-HTML with the following attributes and methods.

#### 8.11.7.2.1.1 IDL Definition

```
interface Navigator {
  readonly attribute String appCodeName;
  readonly attribute String appName;
  readonly attribute String appVersion;
  readonly attribute String userAgent;
};
```

#### 8.11.7.2.1.2 Attributes

**Table 59 : Navigator Object Attributes**

appCodeName	Readonly	Specifies the code name of the browser with the value specified in the BNF in 8.13.1, "User agent strings" on page 163
appName	Readonly	Specifies the name of the browser, the value shall be "DVB-HTML".
appVersion	Readonly	Specifies version information for the Navigator with the value specified in the BNF in 8.13.1, "User agent strings" on page 163
userAgent	Readonly	Specifies the user agent header See 8.13.1, "User agent strings" on page 163

### 8.11.7.2.2 Window object

Represents a top-level browser window or a frame. There is only one top-level window per DVB-HTML application, (i.e. whose parent is NULL). Other window objects in an application represent frames in a frameset.

The top level Window object of an application is created in the CSS initial containing block for the application. The viewport and containing block of the application are controlled by the stylesheet associated with the current document held by the top level Window object. Other Window objects are shaped by the frameset rules.

Window is supported in DVB-HTML with the following attributes and methods.

## 8.11.7.2.2.1 IDL Definition

```

interface Window {
    readonly attribute DVBHTMLDocument document;
    readonly attribute DOMImplementation implementation;
    readonly attribute DVBHTMLCollection frames;
    readonly attribute unsigned long length;
    attribute Location location;
    readonly attribute String name;
    readonly attribute Navigator navigator;
    readonly attribute Window parent;
    readonly attribute Window window;
    readonly attribute Window self;
    attribute String status;
    attribute String defaultStatus;
    readonly attribute Window top;
    void clearTimeout(in unsigned long timerId);
    Window open(in String url, in String name, in String features);
    unsigned long setTimeout(in String statement, in unsigned long delay);
}

```

## 8.11.7.2.2.2 Attributes

The following attributes are supported:

**Table 60 : Window Object Attributes (Sheet 1 of 2)**

Attribute	Modifier	Semantics
closed	Readonly	Specifies whether a window has been closed.
document	Readonly	Contains information on the current document. Returns an object of type DVBHTMLDocument.
implementation	Readonly	Supplies the method to bootstrap into the DOM interfaces.
frames	Readonly	An array reflecting all the frames in a window containing documents from the current application. Frames that belong to another application shall not appear in the collection.
location	Readwrite	Contains information on the URL used to request the current document. Setting the value of window.location sets the href value of the window's associated location object, and causes the window to navigate to the URL. Note that the <b>requested</b> URL is stored in window.location and this may be different than the <b>actual</b> URL of the delivered document, which is stored in document.URL. If a string is used to set the location field, then it is converted to a Location. If the string is not a well formed URL, or the user agent is unable to navigate to it the location is unchanged. Setting this string to the "exit:" locator (see 8.16.5.2, "Exit locator" on page 182) shall cause termination as defined in 9.3.3.5, "Killed" on page 199.
name	Readwrite	A unique name used to refer to this window.
length	Readonly	the length of the frames collection
parent	Readonly	Returns the Window whose frameset contains the current frame or NULL if the window is the outer most window. A frame may return NULL for this attribute for security reasons (see 8.14.3, "Inter application security" on page 172)
status	Readwrite	Specifies a priority or transient message in the window's status bar. (note it is not required this actually be displayed anywhere – but calling it must cause no harm)
defaultStatus	Readwrite	The default message for the status bar (note it is not required this actually be displayed anywhere – but calling it must cause no harm).

**Table 60 : Window Object Attributes (Sheet 2 of 2)**

Attribute	Modifier	Semantics
top	Readonly	A synonym for the topmost browser window. The value of this attribute shall be NULL if the referenced frame is not available for security.
self	Readonly	Returns the current Window
document	Readonly	Returns the document instance the window is displaying. If the document is inaccessible due to security reasons (see 8.14.3, "Inter application security" on page 172) then shall return NULL.
navigator	Readonly	Returns the Navigator object for the application

## 8.11.7.2.2.3 Methods

The following methods are supported:

**Table 61 : Window open method**

Method	Name	Description		
	open	Sets the document of the specified window to a new resource		
Parameters	Name	Type	Qualifier	Description
	url	String	in	The location of the document to display in the new window.
	name	String	in	The target frame within the frameset or the top level window.
	features	String	in	unused
Return value	Type	Description		
	Window	This method returns the Window object in which the document has been opened.		

:

**Table 62 : Window setTimeout method**

Method	Name	Description		
	setTimeout	Evaluates an expression or calls a function once after a specified number of milliseconds has elapsed. When the document is replaced then all timeouts on the window are cleared.		
Parameters	Name	Type	Qualifier	Description
	statement	String	in	The piece of ECMAScript to be evaluated.
	delay	unsigned long	in	The length of time to wait before executing the statement in milliseconds.
Return value	Type	Description		
	unsigned long	This method returns a timerId that uniquely identifies the timer.		



Table 63 : Window clearTimeout method

Method	Name	Description		
	clearTimeout	Cancels a timeout that was set with the setTimeout method. When the document is replaced then all timeouts on the window are cleared		
Parameters	Name	Type	Qualifier	Description
	timerId	unsigned long	in	The id of the timer to clear as returned by setTimeout.
Return value	Type	Description		
	none	This method does not return a value.		

### 8.11.7.2.3 Location object.

A convenience element that parses URIs. Is used in the location field of the window object.

#### 8.11.7.2.3.1 IDL Definition

```
interface Location {
  attribute DOMString hash;
  attribute DOMString host;
  attribute DOMString hostname;
  attribute DOMString href;
  attribute DOMString pathname;
  attribute DOMString port;
  attribute DOMString protocol;
  attribute DOMString search;
};
```

#### 8.11.7.2.3.2 Attributes

Table 64 : Location Object Attributes

Attribute	Modifier	Semantics
hash	Readwrite	The portion of the URI after and including the '#' symbol
host	Readwrite	The hostname and port of the URI
hostname	Readwrite	The hostname (without the port) of the URI.
href	Readwrite	The complete URI.
pathname	Readwrite	The pathname portion of the URI.
port	Readwrite	The port of the URI.
protocol	Readwrite	The protocol portion of the URI.
search	Readwrite	The portion of the URI after and including the '?' symbol, not including the hash portion.

## 8.11.8 CSS Support

### 8.11.8.1 DVB CSS DOM module

DVB-HTML does not require support for the DOM-2 CSS modules, however it does require support of the this simpler CSS module. A DOM application can use the hasFeature method of the DOM implementation interface to determine that this module is supported. The feature string for all interfaces listed in this module is "DVBCSS" and the version "2.0".

#### 8.11.8.1.1 DVBCSSInlineStyle

**DVBInlineStyle:** This interface shall be implemented by objects that implement the Element interface (see [Document Object Model \(DOM\) Level 2 Core Specification \[82\]](#)), and which represent elements that support the style attribute.

A DVB-HTML user agent may support the DOM-2 CSS module, in which case this interface shall be replaced by `ElementCSSInlineStyle`, however in such a case the `CSS2Properties` interface shall also be implemented, and the object returned by the `style` attribute on `ElementCSSInlineStyle` shall also implement the `CSS2Properties` interface

#### 8.11.8.1.1.1 IDL Definition

```
// Introduced in MHP1.1:
interface DVBCSSInlineStyle {
  readonly attribute CSS2Properties style;
};
```

#### 8.11.8.1.1.2 Attributes

Style of type `CSS2Properties` as defined in section 2.3 of [Document Object Model \(DOM\) Level 2 Style Specification \[85\]](#).

#### 8.11.8.1.2 DVBCSSStyle

**DVBCSSStyle:** This interface shall be implemented by objects that implement the `Element` interface (see [Document Object Model \(DOM\) Level 2 Core Specification \[82\]](#)), that represent the root element of a document.

#### 8.11.8.1.2.1 IDL Definition

```
// Introduced in MHP1.1:
interface DVBCSSStyle {
  readonly attribute DVBCSSViewportRule viewport;
};
```

#### 8.11.8.1.2.2 Attributes

`viewport` of type `DVBCSSViewportRule`, If the document containing the element is not in a root window this attribute shall be `NULL`, if it is in the root window then this attribute shall return an object that implements the `DVBCSSViewportRule` interface.

#### 8.11.8.1.3 DVBCSSViewportRule

**DVBCSSViewportRule:** The `DVBCSSviewportRule` interface represents a `@viewport` rule within a CSS style sheet. The `@viewport` rule is used to specify the on screen regions and relationship to video of an DVB-HTML application.

#### 8.11.8.1.3.1 IDL Definition

```
// Introduced in MHP 1.1:
interface DVBCSSViewportRule : CSSRule {
  readonly attribute DVBCSSViewportProperties style;
};
```

#### 8.11.8.1.3.2 Attributes

**style:** of type `DVBCSSViewportProperties`, `readonly`. An object that represents all the properties of the active viewport.

#### 8.11.8.1.4 DVBCSSViewportProperties

**DVBCSSViewportProperties:** The `CSS2Properties` interface represents a mechanism for retrieving and setting properties within a viewport. The attributes of this interface correspond to all the properties specified in [8.8.5.3.2, "The @viewport rule"](#) on page 107.

#### 8.11.8.1.4.1 IDL Definition

```
// Introduced in MHP1.1:
interface DVBCSSViewportRule {
readonly attribute DOMString pseudoClass;
attribute DOMString area;
attribute DOMString[] backgroundVideoTransform;
attribute DOMString[] backgroundVideo;
attribute DOMString[] backgroundVideoPreserveAspect;
attribute DOMString[] backgroundVideoClip;
attribute DOMString[] backgroundVideoRectangle;
attribute DOMString background;
attribute DOMString backgroundImageRectangle;
attribute DOMString initial;
attribute Number verticalResolution;
attribute Number horizontalResolution;
attribute DOMString scene;
};
```

## 8.11.8.1.4.2

## Attributes

**Table 65 : Location Object Attributes**

Attribute	Modifier	Semantics
pseudoClass	Readonly	See the pseudoClass property definition in 8.8.5.3.2, "The @viewport rule" on page 107. Exception: NO_MODIFICATION_ALLOWED_ERR: Raised on any attempt to set this property.
area	Readwrite	See the area property definition in 8.8.5.3.2, "The @viewport rule" on page 107. Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparseable.
backgroundVideoTransform	Readwrite	See the backgroundVideoTransform property definition in 8.8.5.3.2, "The @viewport rule" on page 107. Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparseable. (note 1)
backgroundVideo	Readwrite	See the backgroundVideo property definition in 8.8.5.3.2, "The @viewport rule" on page 107. Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparseable. (note 1)
backgroundVideoPreserveAspect	Readwrite	See the backgroundVideoPreserveAspect property definition in 8.8.5.3.2, "The @viewport rule" on page 107. Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparseable. (note 1)
backgroundVideoClip	Readwrite	See the backgroundVideoClip property definition in 8.8.5.3.2, "The @viewport rule" on page 107. Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparseable. (note 1)
backgroundVideoRectangle	Readwrite	See the backgroundVideoRectangle property definition in 8.8.5.3.2, "The @viewport rule" on page 107. Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparseable. (note 1)
background	Readwrite	See the background property definition in 8.8.5.3.2, "The @viewport rule" on page 107. Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparseable. (note 1)
backgroundImageRectangle	Readwrite	See the backgroundImageRectangle property definition in 8.8.5.3.2, "The @viewport rule" on page 107. Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparseable. (note 1)
initial	Readwrite	See the initial property definition in 8.8.5.3.2, "The @viewport rule" on page 107. Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparseable.
verticalResolution	Readwrite	See the verticalResolution property definition in 8.8.5.3.2, "The @viewport rule" on page 107. Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparseable.
horizontalResolution	Readwrite	See the horizontalResolution property definition in 8.8.5.3.2, "The @viewport rule" on page 107. Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparseable.
NOTE 1: Note this attribute is an array of values, one for each defined background plane		

## 8.12 Cookie support

### 8.12.1 DOM Cookie Interface

The cookie attribute of the document (see section 8.11.4.5.2, "DVBHTMLDocument Interface" on page 138) provides an interface to the set of attribute-value pairs described in RFC 2109 [96].

The cookie attribute of document is a read/write string with the following behaviour:

- a) The attribute may be assigned a string value `set-cookie` obeying the following BNF, where ALPHA and DIGIT are as specified in RFC 2616 [40]:

```
set-cookie= cookie
cookie= av-pair
      ["; expires=" [ rfc1123-date ] ]
      ["; path=" [ path ] ]
      ["; domain=" [ domain] ]
      ["; secure" ]
      ["; dvb-scope=" [ dvb-scope ] ]
av-pair= attr ["=" value ]
attr = 1*(ALPHA | DIGIT | aspecials)
aspecials= ( "!" | "#" | "$" | "&" | "'" | "*" | "+" | "-"
            | "." | "_" | "`" | "|" | "~" | "%" )

value = 1*(ALPHA | DIGIT | special | vspecials)
vspecials = "(" | ")" | "<" | ">" | "@" | "," | ":" | "\" | "<" | "/"
            | "[" | "]" | "?" | "=" | "{" | "}" | SP | HT

path = 1*(ALPHA | DIGIT | aspecials)
domain = 1*(ALPHA | DIGIT | aspecials)
dvb-scope = "App" | "Org" | "All"
```

rfc1123-date is the subset of the RFC 1123 [97] date format as specified in RFC 2616 [40] (HTTP 1.1).

```
rfc1123-date = wkday " ", " SP date1 SP time " GMT"
wkday       = "Mon" | "Tue" | "Wed" | "Thu" | "Fri" | "Sat" | "Sun"
date1      = 2DIGIT SP month SP 4DIGIT
time       = 2DIGIT ":" 2DIGIT ":" 2DIGIT
month      = "Jan" | "Feb" | "Mar" | "Apr" | "May" | "Jun"
            | "Jul" | "Aug" | "Sep" | "Oct" | "Nov" | "Dec"
```

- b) If an expiration is not specified, the lifetime of the cookie is that of the DVB-HTML application.
- c) When read the attribute returns a string `get-cookie` consisting of the av-pairs for all cookies that have not expired and are within the scope of the current document.

```
get-cookie = av-pair *(";" av-pair)
```

Note: If the same attribute is specified for multiple paths that apply to the current document, the attribute appears in multiple av-pairs.

Note: "%" is a legal character in values. As defined in RFC 2109 [96] characters not allowed in values may be escaped using "% HEX HEX", there is no special handling of this escaping by the ECMAScript API.

### 8.12.2 Cookie Storage and Lifetime

#### 8.12.2.1 Cookie Storage Limits

An implementation should attempt to meet the minimum implementation limits of RFC 2109 [96] including (although not necessarily simultaneously) at least 20 cookies of 4096 bytes. The size is the total required for the cookies and all attributes stored as strings.

#### 8.12.2.2 Cookie Persistence

If the application requests permission for access to persistent storage, this is granted and persistent storage is available, an implementation should attempt to store cookies until they expire. If the application does not have permission for persistent storage or the storage is not available then cookies shall persist only for the lifetime of the DVB-HTML application.

### 8.12.2.3 Privacy Considerations

Notwithstanding any other requirements in this section, an implementation may provide mechanisms for controlling the extent of cookie support, including which, if any, cookies are stored, temporarily or persistently.

## 8.12.3 Cookie Scoping

### 8.12.3.1 General Rules

All cookies are scoped by `dvb-scope`. The `dvb-scope` limits cookies to being accessible by pages in applications of a particular organization (`dvb-scope=Org`), by pages with a particular Application Identifier (`dvb-scope=App`) or by pages within all applications (`dvb-scope=All`). When not specified, the `dvb-scope` defaults to `App`. Cookies of unsigned applications are always scoped to `App`. Cookies with different scoping are stored separately and reported separately both through the DOM and when transmitted to a server. Setting a cookie with the same scoping as an existing one causes the replacement of the existing one.

### 8.12.3.2 Documents delivered via DSM-CC Object Carousel

Cookies set via the DOM on pages delivered by DSM-CC carousel transport are never sent back to a server, even when `http: URLs` are within the application boundary. The domain, path and secure attributes are ignored for such cookies.

### 8.12.3.3 Documents delivered via HTTP transport

Cookies associated with pages delivered by `http:` shall be scoped by domain and path in addition to scoping by their `dvb-scope`. If the domain is specified, the cookie is only accessible to pages delivered from hosts within that domain. The default domain is the host name of the `http` server which delivered the page.

If the path is specified, the associated cookie is only accessible to pages in directories under that path.

Domain and path matching are performed per [RFC 2109 \[96\]](#) with the exception that a host match is generated if either the domain string or the domain string with a period prepended to it would generate a match with the basic algorithm.

NOTE: the modified matching algorithm provides both domain compatibility with cookies whose domain is set from the DOM or from HTTP 1.0 or 1.1 servers that only support Netscape cookie protocol. In both cases, leading dots are frequently left off.

Setting a cookie with a domain or path for which the current page is out of scope has no effect and the cookie is discarded.

Cookies with the secure attribute specified shall only be sent over secure, HTTPS connections.

## 8.12.4 HTTP Cookie Support

### 8.12.4.1 Background

The [MHP profile of HTTP 1.0](#) profile allows both HTTP 1.0 and HTTP 1.1 transport (see 6.3.7.2, "[MHP profile of HTTP 1.0](#)" on page 58). In order for cookie transport to work with existing HTTP 1.0 servers, support for "Netscape" cookie receipt is required. Cookie transmission can always be done with [RFC 2109 \[96\]](#) format since most servers will consider the `$Version` and `$Path` attributes as unknown cookies and ignore them.

### 8.12.4.2 Sending Cookies

The client shall send cookies per [RFC 2109 \[96\]](#) and for compatibility with older servers, DVB-HTML implementations should use semi-colons to separate cookies rather than commas (both are allowed by [RFC 2109 \[96\]](#)). Only cookies in the scope of the document, including any `dvb-scope` restrictions, are sent.

### 8.12.4.3 Receiving Cookies

The client shall accept cookies via a `Set-Cookie` header in the HTTP response. The format of the contents of that header may be either in [RFC 2109 \[96\]](#) format or in "Netscape" cookie format, where Netscape format is that specified by the `set-cookie` BNF from the DOM description above.

## 8.13 HTTP User Agent String Support

Requests from the DVB-HTML user agent shall include an HTTP header of the form:

User-Agent: <user-agent>

where <user-agent> is the DVB-HTML user agent string.

### 8.13.1 User agent strings

#### 8.13.1.1 Current user agent-related strings

Current examples (from the internet) of the user agent string are shown in the following table.

**Table 66 : Internet user agent string examples**

Field	Netscape Navigator	Internet Explorer on Win2K
userAgent	Mozilla/4.7 [en] (WinNT; U)	Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
appName	Mozilla	Mozilla
appVersion	4.7 [en] (WinNT; U)	4.0 (compatible; MSIE 5.01; Windows NT)
appName	Netscape	Internet Explorer

The last three of these appear as properties on the navigator object. The properties `appName` and `appVersion` are derived from the user agent string as reflected in the BNF. The user agent string generally has the BNF format. The `appName` is independent of the user agent string and its value shall always be "DVB-HTML" (see 8.11.7.2.1, "Navigator Object" on page 154).

#### 8.13.1.2 User agent string BNF

Adopting "Mozilla" as the `appName` provides no interoperability benefit, and in fact would be incorrect as the DVB HTML user agent is not required to be Mozilla compliant. In general, the user agent string is best kept short, since it is sent with every HTTP request, hence it is reasonable to use abbreviations where appropriate.

The following BNF illustrates the user agent string that should be used:

```

userAgent ::= appCodeName "/" appVersion
appCodeName ::= "DVB-HTML"
appVersion ::= appVersionNumber " (" identifiers ")"
appVersionNumber ::= major "." minor "." micro
major ::= [0-9]+
minor ::= [0-9]+
micro ::= [0-9]+
identifiers ::= [ profileIdentifier ]+ [ ";" " mhpIdentifier ]*
profileIdentifier ::= profileName " " major "." minor "." micro
profileName ::= "eb" | "ib" | "ia"
mhpIdentifier ::= [^;()]+

```

The syntax for the `major`, `minor` and `micro` elements are equal to the properties with the same names defined in 11.9.3, "Profile and version properties" on page 270. The values for these properties are specified in 16.1, "System constants" on page 365. `mhpIdentifiers` may be used to indicate optional features in the MHP implementation.

For example:

```
DVB-HTML/1.1.0 (eb 1.1.0; ipm)
```

Indicates an enhanced broadcast profile, with optional ip multicast support.

## 8.14 Security of DVB-HTML applications

### 8.14.1 Authentication of DVB-HTML files

DVB-HTML is subject to the same security model as DVB-J applications, that is a permissions file is associated with authenticated applications; unauthenticated applications operate within a sandbox environment, which means certain forms of locators will not operate, and certain APIs will not be available to ECMAScript.

Authenticated applications may be granted permission to use the restricted locators and APIs.

User Agents shall interpreting signed applications shall only make available the following file types to that application if they are signed by the same leaf certificate:

- DVB-HTML
- ECMAScript
- CSS
- Inner application files (for DVB-J class files see 11.2.3, "Class Loading" on page 241)

Locators that are disabled due to security restrictions will behave as if the resource pointed to is unavailable.

ECMAScript API calls that fail due to security restrictions will generate the [DVBException](#) with the error code [SECURITY\\_VIOLATION\\_ERR](#).

### 8.14.2 Runtime code extension

#### 8.14.2.1 Security principles

In order to address the security issues that arise with runtime code extension:

- a) ECMAScript platforms shall permit the use of any of the ECMAScript runtime code extension techniques listed below for unauthenticated applications or authenticated applications running in the sandbox
- b) ECMAScript platforms shall permit the restricted use of runtime code extension techniques for authenticated applications running outside of the sandbox, where by restricted use is meant that the input to runtime code extension derives only from an appropriately authenticated source (see ["Extensions to ECMAScript for trusted executable code"](#) on page 165)
- c) ECMAScript platforms shall not permit the removal of pre-defined properties of host objects;
- d) ECMAScript platforms shall not permit the replacement of a pre-defined function property of a host object.

##### 8.14.2.1.1 Uses of runtime code extension in ECMAScript

The ECMAScript language supports runtime code extension by means of the following separate mechanisms:

- the `eval()` function property of the global object;
- the `Function` object.

In addition, the following mechanisms support runtime code extension through the following host object (platform) facilities:

- the `write()` function property of the DVB-HTML Document host object.
- the `setTimeout(in String statement, in unsigned long delay)` function property of the Window object.

The rules covering runtime code extension are detailed in 8.14.2.4, ["Use of strings in RCEs"](#) on page 171



## 8.14.2.2 Extensions to ECMAScript for trusted executable code

This section details DVB specific mechanisms to allow Runtime Code Extension (RCE) features specified by the ECMAScript language specification and the allowed DOM APIs to be used without compromising security. In DVB-HTML additional mechanisms that keep track of the origin of strings in the system and disallow the potentially dangerous use in RCEs by privileged applications of strings from unverified external sources.

### 8.14.2.2.1 Propagation of Internal (safe) vs. External (unsafe) strings

This approach to providing security for the use of ECMAScript maintains application compatibility by distinguishing between value of an internal string type, a string value coming from within the ECMAScript language or literal strings within the application, and an external string type, a string value coming from (suspect) host-based objects. Both string values appear to content as ordinary ECMAScript string values to improve compatibility with the behaviour of DVB-HTML content in standard Internet browsers. This is accomplished by modifying the ECMAScript language specification to split the string type into two internal subtypes: *internal-string* and *external-string*. The difference between these subtypes is not visible to the application programmer, except in so far as a security exception may be thrown as a result of the use of some strings in an RCE.

This approach has the advantage of maintaining content compatibility with ECMAScript content written for existing browsers and to the W3C and ECMA specifications. In particular,

- a) It allows DVB-HTML to conform to the W3C specified DOM APIs by allowing them to return values of type String rather than of type String Object.
- b) It minimizes any increase in the runtime data footprint of ECMAScript applications by allowing the continued use of the much more compact String type, rather than String Object.
- c) It minimizes any implementation change for DOM bindings between a W3C-complaint version and a DVB-HTML compliant version, since both can return String types.
- d) It eliminates compatibility problems that would otherwise require changing core ECMAScript behaviour if String Objects were used (e.g. that two different instances of an object never test true for equality).
- e) It allows finer-grained propagation of unsafe typing, since every string entity in the ECMAScript runtime would be tagged as safe (internal) or unsafe (external).

### 8.14.2.2.2 Modifying ECMA-262 to support Internal and External strings

This section normatively specifies modifications to [ECMAScript \[81\]](#) to differentiate the two kinds of string values.

This document uses the editing convention that additions to [ECMAScript \[81\]](#) are given in *italics*. Text that is removed is presented as ~~strike-out~~. When a change is made, the old text is given as ~~strike-out~~ and the new version in *italics*. The bold and bold-italics font faces are preserved from the ECMAScript standard document.

Change the last sentence of the second paragraph of section 4.2 to read:

A primitive value is a member of one of the following built-in types: **Undefined, Null, Boolean, Number, and String** *Internal-String, and External-String*; an object is a member of the remaining built-in type **Object**; and a method is a function associated with an object via a property. *The term String type is used when either of the types Internal-String or External-String is acceptable. The term String value is used when a value of either of the types Internal-String or External-String is acceptable.*

Change the first sentence of the first paragraph of section 4.3.2 to read:

A *primitive value* is a member of one of the types **Undefined, Null, Boolean, Number, and String** *Internal-String, and External-String*.

Change sections 4.3.16 through 4.3.18 to read:

**4.3.16 String Value** *String, Internal-String, and External-String Values*

*4.3.16.1 String Value*

A *string value* is a member of the type ~~String~~ and either a member of the type Internal-String or a member of the type External-String. It is a finite ordered sequence of zero or more 16-bit unsigned integer values.

NOTE: Although each value usually represents a single 16-bit unit of UTF-16 text, the language does not place any restrictions or requirements on the values other than that they be 16-bit unsigned integers.

#### 4.3.16.2 Internal-String Value

An internal-string value is a member of the type Internal-String. See section 4.3.16.1 for details.

#### 4.3.16.3 External-String Value

An external-string value is a member of the type External-String. See section 4.3.16.1 for details.

### 4.3.17 ~~String Type~~ String, Internal-String, and External-String Types

#### 4.3.17.1 String Type

The type **String** is the set of all string values.

#### 4.3.17.2 Internal-String Type

The type Internal-String is the set of all internal-string values.

#### 4.3.17.3 External-String Type

The type External-String is the set of all external-string values.

In section 4.8.4, change the second paragraph after the BNF to read:

A string literal stands for a value of the ~~String~~ *Internal-String* type. The ~~string~~ *internal-string* value (SV) of the literal is described in terms of the character values (CV) contributed by the various parts of the string literal. As part of this process, some characters within the string literal are interpreted as having a mathematical value (MV), as described below or in section 7.8.3.

Change the first line of Section 8.4 to read:

### 8.4 The ~~String Type~~ *Internal-String* and *External-String* Types

Insert the following immediately before section 8.5:

#### 8.4.1 The Internal-String Type

The Internal-String type is used for strings that originate purely from string literals within an ECMAScript program, from conversions of non-external-string types to a string type, and from the methods of built-in objects other than String object. The methods of the String object will return either Internal-String or External-String values, depending on the type of the string used to initialize the String object.

#### 8.4.2 The External-String Type

The External-String type is used for strings that are returned from host objects.

In section 9.1, change the table row for Input Type String into the following two table rows:

**Table 1:**

<del>String</del> <i>Internal-String</i>	The result equals the input argument (no conversion)
External-String	The result equals the input argument (no conversion)

In section 9.2, change the table row for Input Type String into the following two table rows:

**Table 2:**

<del>String</del> Internal-String	The result is <b>false</b> if the argument is the empty string (its length is zero); otherwise the result is <b>true</b> .
External-String	The result is false if the argument is the empty string (its length is zero); otherwise the result is true.

In section 9.3, change the table row for Input Type String into the following two table rows:

**Table 3:**

<del>String</del> Internal-String	See grammar and note below.
External-String	See grammar and note below.

Change the first line of section 9.3.1 to read:

**9.3.1 ToNumber Applied to the ~~String Type~~ Internal-String and External-String Types**

In section 9.8, change the table to read:

**Table 4:**

Input Type	Result
Undefined	<i>the Internal-String value "undefined"</i>
Null	<i>The Internal-String value "null"</i>
Boolean	If the argument is <b>true</b> , then the result is <i>the Internal-String value "true"</i> .  If the argument is <b>false</b> , then the result is <i>the Internal-String value "false"</i>
Number	See note below
<del>String</del> Internal-String	Return the input argument (no conversion)
External-String	Return the input argument (no conversion)
Object	Apply the following steps:  Call ToPrimitive(input argument, hint String). Call ToString(Result(1)). Return Result(2).

Change the first paragraph under section 9.8.1 to read:

The operator ToString converts a number *m* to ~~string format~~ *an internal-string value* as follows:

Immediately before section 10, insert:

9.10 ToInternalString

The operator ToInternalString converts its argument to a value of type Internal-String according to the following table:

**Table 5:**

Input Type	Result
Undefined	the Internal-String value "undefined"
Null	The Internal-String value "null"

**Table 5:**

Boolean	If the argument is true, then the result is the Internal-String value "true". If the argument is false, then the result is the Internal-String value "false"
Number	See note in section 9.8
Internal-String	Return the input argument (no conversion)
External-String	if CanConvertToInternal(argument) returns true, then the result is an internal string with the same sequence of (16-bit) character values.  Otherwise, throw SecurityError.
Object	Apply the following steps:  Call ToPrimitive(input argument, hint String). Call ToString(Result(1)). If CanConvertToInternal(Result(2)) returns true, return Result(2). Otherwise, throw SecurityError.

The operator CanConvertToInternal is implemented so as to return false if the ECMAScript is being executed in a trusted context and no permissions have been granted to the context to convert external strings to internal strings. In all other cases, the operator CanConvertToInternal will return true.

Change section 10.1.2 to read:

There are three types of ECMAScript executable code:

- Global code is source text that is treated as an ECMAScript Program. The global code of a particular Program does not include any source text that is parsed as part of a FunctionBody.
- Eval code is the source text supplied to the built-in **eval** function. More precisely, if the parameter to the built-in **eval** function is ~~a string~~ *an internal-string*, it is treated as an ECMAScript Program. The eval code for a particular invocation of **eval** is the global code portion of the ~~string~~ *internal-string* parameter.
- Function code is source text that is parsed as part of a FunctionBody. The function code of a particular FunctionBody does not include any source text that is parsed as part of a nested FunctionBody. Function code also denotes the source text supplied when using the built-in **Function** object as a constructor. More precisely, the last parameter provided to the **Function** constructor is converted to ~~a string~~ *an internal-string* and treated as the FunctionBody. If more than one parameter is provided to the **Function** constructor, all parameters except the last one are converted to strings and concatenated together, separated by commas. The resulting string is interpreted as the FormalParameterList for the FunctionBody defined by the last parameter. The function code for a particular instantiation of a **Function** does not include any source text that is parsed as part of a nested FunctionBody.

In section 11.4.3, change the single table row for Type String into the following two table rows:

**Table 6:**

<del>String</del> <i>Internal-String</i>	"string"
External-String	"string"

In section 11.6.1, change steps 14 and 15 to read:

~~14. Concatenate Result(12) followed by Result(13).~~

~~15. Result Result(14)~~

14. If Result(12) is of type *internal-string* and Result(13) is of type *external-string*, use a string value with the same sequence of characters as Result(12) but of type *external-string*. Otherwise use Result(12).

15. If Result(12) is of type external-string and Result(13) is of type internal-string, use a string value with the same sequence of characters as Result(13) but of type external-string. Otherwise use Result(13).

16. Concatenate Result(14) followed by Result(15). The string type is the same as that of Result(14).

17. Return Result(16).

Append the following paragraph at the end of Section 12.6.4:

Property names are maintained as internal-strings or external-strings depending on the source of the property name. Property names are returned as internal-strings unless the property name was set from an external string, in which case it is returned as an external string.

Change step 1 under section 15.1.2.1 to read:

1. If x is not a string value, return x. *If x is an external-string value and CanConvertToInternal(x) returns false, throw the exception SecurityError.*

In section 15.1.3, change step 4 in the procedure for the hidden function Encode to read:

4. If k equals Result(1), return ~~R~~ *a string with the same sequences of characters as R and the same (internal-string vs external-string) type as the input string.*

In section 15.1.3, change step 4 in the procedure for the hidden function Decode to read:

4. If k equals Result(1), return ~~R~~ *a string with the same sequences of characters as R and the same (internal-string vs external-string) type as the input string.*

In section 15.2.4.2, change step 2 to read:

2. Compute ~~a string~~ *an internal-string* value by concatenating the three strings "[object ", Result(1), and "]".

In section 15.3.2.1, change step 13 to read:

13. Call ~~ToString~~ *ToInternalString*(body)

In section 15.4.4.3, change step 14 to read:

Let R be a string value produced by concatenating S and Result(13). *The type of R is external-string if the type of either S or Result(13) is external-string. Otherwise, the type of R is internal-string.*

In section 15.4.4.5, change step 11 to read:

Let S be a string value produced by concatenating R and Result(4). *The type of S is external-string if the type of either R or Result(4) is external-string. Otherwise, the type of S is internal-string.*

In section 15.4.4.5, change step 14 to read:

Let R be a string value produced by concatenating S and Result(13). *The type of R is external-string if the type of either S or Result(13) is external-string. Otherwise, the type of R is internal-string.*

Change the first sentence of section 15.5.3.2 to read:

Return ~~a string~~ *an external-string* value containing as many characters as the number of arguments.

Add the following sentence to the end of the first paragraph in section 15.5.4.2 and to the first paragraph in section 15.5.4.3:

*Return the value as an external-string if the String object was constructed with a value that converted to an external-string. Return the value as an internal-string if the String object was constructed with a value that converted to an internal-string.*

Add the following sentence after the first paragraph of section 15.5.4.4 to read:

*Returns an external-string, if the type of the result of ToString is external-string. Otherwise, the returned string is an internal-string.*

Change step 5 in section 15.5.4.6 to read:

Let R be the string value consisting of the characters in the previous value of R followed by the characters Result(4). *If either the previous value of R or Result(4) is of type external-string, give R type external-string. Otherwise give R type internal-string.*

Insert the following paragraph just before the NOTE in section 15.5.4.1:

*If either the result of converting this to a string is of type external-string or the replaceValue converted to a string is of type external-string or the function replaceValue returns a value of type external-string, then the newstring has type external-string. In all other cases, the newstring has type internal-string.*

In section 15.5.4.13 to read, change step 8 to read:

8. Return a string containing Result(7) consecutive characters from Result(1) beginning with the character at position Result(5). *The type of the returned string is external-string if the type of Result(1) is external-string. The type of the returned string is internal-string if the type of Result(1) is internal-string.*

In section 15.5.4.14, change step 16 to read:

16. Let T be a string value equal to the substring of S consisting of the characters at positions p (inclusive) through q (exclusive). *Let the type of T (internal-string versus external-string) be the same as the type of S.*

In section 15.5.4.14, change step 28 to read:

28. Let T be a string value equal to the substring of S consisting of the characters at positions p (inclusive) through q (exclusive). *Let the type of T (internal-string versus external-string) be the same as the type of S.*

In section 15.5.4.15, append the following to the end of step 9:

*Let the type of the returned string (internal-string versus external-string) be the same as that of Result(1).*

In section 15.5.4.16, insert the following paragraph before NOTE1:

The result string's type (internal-string vs external-string) is the same as that of the string computed by converting the object to a string.

In sections 15.9.5.2 through 15.9.5.7, change the first sentence of the first paragraph in each section to read:

This function returns ~~a string~~ *an internal-string* value.

In section 15.10.6.2, append the following at the end section::

*The type of the resulting string (internal-string vs external-string) is the same as that of the string argument.*

In section 15.10.6.4, append the following at the end of the second paragraph:

The returned string is of type internal-string if source is an internal-string. The returned string is of type external-string if source is an external-string.

Insert the following immediately after Section 15.1.6.6:

#### 15.1.6.7 SecurityError

Indicates that an authorized attempt was made to convert an external-string into an internal-string within the context of trusted ECMAScript.

### 8.14.2.3 Sources of Unsafe (external) strings

#### 8.14.2.3.1 Sources within ECMAScript

As specified in the previous section the following sources of unsafe strings within the language itself shall be typed as external strings:

- a) Strings resulting from operations on external-strings
- b) `String.fromCharCode`, which converts a character code into a string.

#### 8.14.2.3.2 Sources from Host Objects

All strings returned by Host Object APIs defined as part of this specification shall be typed as external strings.

All strings returned by the Bridge shall be typed as external strings.

Any Host Object APIs provided by a platform beyond those required in this specification should return external strings unless the result is known to be secure.

### 8.14.2.4 Use of strings in RCEs

An attempt to use an external string or a String Object containing an external string in a runtime code extension shall cause the `DVBException` with the error code `SECURITY_VIOLATION_ERR` to be raised.

An attempt to use an internal string or a String Object containing an internal string in a runtime code extension shall not cause the `DVBException` with the error code `SECURITY_VIOLATION_ERR` to be raised.

If the permission request file requests the permission to do privileged runtime code extension and this is granted for external strings then use of an external string or a String Object containing an external string in a runtime code extension shall be considered use of an internal string or a String Object containing an internal string.

If no permission to do privileged runtime code extension is granted then any attempt to use an internal or external string or a String Object containing an internal or external string in a runtime code extension shall be considered use of an external string.

If the permission request file requests the permission to do privileged runtime code extension and this is granted for internal strings only then use of external and internal strings retain their type.

The following uses of a string shall be considered runtime code extensions:

A string:

- a) passed to `eval()`
- b) passed to the constructor of a `Function()` object
- c) passed to `document.write()`
- d) used to set an event handler.

The exception occurs when an external-string is associated with the event handler either by setting a handler attribute of an element in the document or by parenting

- e) used as the code in a script element.

The setting of a `<script>` element text or `src` attribute in a displayed document shall not cause the new text to be incorporated into the ECMAScript context for the document. Therefore this is not strictly runtime code exception. However modification of script nodes is disallowed in this specification. The exception occurs when the external-string is associated with the script element either by parenting to a script element a node containing an external-string (e.g. by `Node::insertBefore()`, `Node::appendChild()`, `Node::replaceChild()`) or by setting the `nodeValue` of a `TextNode` or a `CDATASection` child of a script element to an external string.

- f) used in `window.setTimeout()`

### 8.14.2.5 Mutation of Host Objects

Any attempt to remove or replace the pre-defined properties of a host object shall cause the `DVBException` with the error code `SECURITY_VIOLATION_ERR` to be raised.

Any attempt to remove or replace the pre-defined function property of a host object shall cause the `DVBException` with the error code `SECURITY_VIOLATION_ERR` to be raised.

## 8.14.3 Inter application security

### 8.14.3.1 Restrictions on DOM elements introduced for security

No access is provided to documents from other applications.

The property accessor on the window and document host objects shall restrict access to properties and functions of those objects when the document context in which the access is executed is from a different DVB-HTML application or from a different domain than the document in the window:

- For such contexts, the window accessor shall only allow the assignment of a string to the location property: setting any other window property or getting any window property shall throw an exception.
- For such contexts, the document accessor shall not allow any access: setting or getting any document property shall throw an exception.

The exception that is thrown shall be the appropriate exception as if an attempt to access a resource for which the caller did not have permission were made:

- in the Java binding (see annex [AD.1](#), "[Java bindings to DVB extensions](#)" on page 1296) a `SecurityException` shall be thrown
- in the ECMAScript binding (see annex [AC](#), "[\(normative\): ECMAScript Binding](#)" on page 1283) when a security exception is raised, the `DVBException` with the error code `SECURITY_VIOLATION_ERR` is raised.

If a new application is started in a frame, then a new Window object is created to represent the frame. The parent of this window is the window that represents the frame from the calling applications perspective. This parent property however is not visible to the application within the frame. Therefore both applications have a window object that controls the frame, but no communication is possible between them. Access to properties, methods, and variables involving Window objects are restricted to windows that are displaying documents from the same application and from the same domain (see [8.16.6](#), "[Domain](#)" on page 182).

Where an application within a frame terminates prior to the application which contains the frameset that hosts the frame for that application, then the latter application shall receive the DVB lifecycle event "`AppTerminating`" (see "[Lifecycle events](#)" on page 129).

The user agent shall set the content of the frame that contained the terminating application to a blank document. and the URL and Location property shall be set to the null string prior to sending the "`AppTerminating`" event.

Where an outer application terminates with a frameset which contains one or more embedded applications in frames, then the applications in those frames shall also be terminated. This shall apply even in the case where the application within the sub frame would survive service selection but the outer application will not.

## 8.15 DVB-HTML permissions

This section explains how the permissions that define the sandbox and are available to unsigned applications and the permissions that can be requested in the permission request file are mapped to the DVB-HTML application model. ADVB-HTML application has the following ways of accessing beyond the sandbox.

- a) Following embedded links in the page can cause the loading of new resources.
- b) Script may access APIs over the Java Bridge.



c) Properties in the host objects available to script may access resources.

The location attribute of the Window object.

The open() Function attribute of the Window object.

d) <uri()> references in CSS attributes

In case c) setting of attributes via the DOM interfaces is handled only when the attribute is used, and in this case the restrictions of case a) are applied.

In some cases a permission is only relevant to APIs reached via the Java Bridge

## 8.15.1 Permissions for unsigned applications

The MHP security policy includes a set of resources that are always guaranteed to be granted to applications, if the application is executed. Unsigned applications have access to only these resources.

This section defines the mapping of those resources to DVB-HTML applications. DVB-HTML applications shall always be granted the necessary Permission.

### 8.15.1.1 java.awt.AWTPermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.1.1, "[java.awt.AWTPermission](#)" on page 272.

### 8.15.1.2 java.net.SocketPermission:

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.1.2, "[java.net.SocketPermission](#):" on page 272.

### 8.15.1.3 java.util.PropertyPermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.1.3, "[java.util.PropertyPermission](#)" on page 272.

### 8.15.1.4 java.lang.RuntimePermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.1.4, "[java.lang.RuntimePermission](#)" on page 272.

### 8.15.1.5 java.io.SerializablePermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.1.5, "[java.io.SerializablePermission](#)" on page 272.

### 8.15.1.6 java.io.FilePermission

Permission to read resources for the sub-tree under which the implementation mounts the object carousels shall be granted. For APIs accessed via the, the same rights are granted as for DVB-J applications, see 11.10.1.6, "[java.io.FilePermission](#)" on page 273.

The location attribute of the Window object may be set to a resource in the sub-tree under which the implementation mounts the object carousels using a URI with the "dvb:" protocol.

The open() Function attribute of the Window object may be called with reference to a resource in the sub-tree under which the implementation mounts the object carousels using a URI with the "dvb:" protocol.

DVB HTML can read resources for the sub-tree under which the implementation mounts the object carousels using the dvb: protocol in a URI in the following element attributes:

- a.href
- area.href
- base.href
- blockquote.cite
- frame.longdesc
- frame.src
- iframe.longdesc
- iframe.src
- img.src
- img.longdesc
- img.usemap
- input.src
- input.usemap
- link.href
- object.archive
- object.classid
- object.codebase
- object.data
- object.usemap
- q.cite
- script.src
- Using the CSS uri() value

Using a "dvb:" locator that references a resource in the Carousel on other element attributes is undefined.

#### 8.15.1.7 javax.tv.media.MediaSelectPermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.1.7, "[javax.tv.media.MediaSelectPermission](#)" on page 273.

#### 8.15.1.8 javax.tv.service.ReadPermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.1.8, "[javax.tv.service.ReadPermission](#)" on page 273.

#### 8.15.1.9 javax.tv.service.selection.ServiceContextPermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.1.9, "[javax.tv.service.selection.ServiceContextPermission](#)" on page 273.

### 8.15.1.10 java.util.Locale.setDefault

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.1.10, "java.util.Locale.setDefault" on page 273.

### 8.15.1.11 org.dvb.security.PrivilegedRCEPermission

Unsigned applications are granted PrivilegedRCEPermission("external"), see "Use of strings in RCEs".

## 8.15.2 Additional Permissions for signed applications

Signed applications can additionally request to get more permissions. These permissions are requested using the permission request file (12.6, "Security policy for applications" on page 297). This section defines the mapping from the items in the permission request file to the permissions that may be granted by the MHP terminal in response to the request. And how this affects the DVB-HTML application

In the case of org.dvb.security.PrivilegedRCEPermission only, signed applications with a permission request file can request permission that is more restrictive than that granted to applications without a permission request file.

### 8.15.2.1 java.util.PropertyPermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.2.1, "java.util.PropertyPermission" on page 273.

### 8.15.2.2 java.io.FilePermission

A signed application has the rights of an unsigned application, in addition when the permission request file requests the permission to access persistent storage and this is granted, read and write access to the persistent storage directory subtree is allowed.

This Permission is relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.2.2, "java.io.FilePermission" on page 273.

If the permission is granted, the location attribute of the Window object may be set to a resource in the persistent file store using a URI with the "file:" protocol. If the permission is not granted then attempting to set this attribute with such a locator will cause a SecurityException to be thrown.

If the permission is granted, the open() Function attribute of the Window object may be called with reference to a resource in the persistent file store using a URI with the "file:" protocol. If the permission is not granted then calling open() with such a locator will cause a SecurityException to be thrown.

DVB HTML can read the persistent file store using the file: protocol in a URI in the following element attributes:

- a.href
- area.href
- base.href
- blockquote.cite
- frame.longdesc
- frame.src
- iframe.longdesc
- iframe.src
- img.src
- img.longdesc
- img.usemap
- input.src
- input.usemap
- link.href
- object.archive
- object.classid
- object.codebase
- object.data
- object.usemap
- q.cite
- script.src
- Using the CSS uri() value

If the file permission is not granted, all "file:" references shall fail as if the referenced resource was not available.

Using a "file:" locator that references a resource in the persistent store on other element attributes is undefined.

When there is a persistent file credential in the permission request file and this is granted, other files are made available as follows:

- file path = root directory of the persistent storage + filename from the credential
- action = string containing "read" and/or "write" as indicated in the credential

### 8.15.2.3 org.dvb.net.ca.CAPermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.2.3, "[org.dvb.net.ca.CAPermission](#)" on page 273.

### 8.15.2.4 org.dvb.application.AppsControlPermission

This Permission is relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.2.4, "[org.dvb.application.AppsControlPermission](#)" on page 273.

If the permission is granted, the location attribute of the Window object may be set to launch a DVB-HTML application in a sub-window by using a locator which references another DVB-HTML application. If the permission is not granted or the locator points to an application of a type which is not embeddable in a frame then attempting to set this attribute with such a locator will cause a SecurityException to be thrown.

If the permission is granted, the open() Function attribute of the Window object may be called with reference to a resource that launches a DVB-HTML application in a sub-window by using a locator which references another DVB-HTML application. If the permission is not granted or the locator points to an application of a type which is not embeddable in a frame then calling the open() with such a locator will cause a SecurityException to be thrown.

DVB-HTML applications can, if granted this permission, launch applications using locators that reference other applications (see 9.3.1.4, "Application boundary" on page 194 and 9.2.1, "Starting DVB-J Applications" on page 187) using the attributes on elements below:

- a.href
- area.href

If the application is not granted the permission then these links will fail as if the resource was not available.

Using a locator that references an application on other element attributes is not required to be supported.

### 8.15.2.5 org.dvb.net.rc.RCPermission

This Permission is relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.2.5, "org.dvb.net.rc.RCPermission" on page 274.

If the permission is granted for the default ISP, setting the location attribute of the Window object using locators that would require creating a return channel link may establish the link required to access the resource. If the permission is not granted then attempting to set this attribute with such a locator will cause a SecurityException to be thrown.

If the permission is granted for the default ISP, using the open() Function attribute of the Window object using locators that would require creating a return channel link may establish the link required to access the resource. If the permission is not granted then calling open() with such a locator will cause a SecurityException to be thrown.

DVB-HTML applications can, if granted this permission for the default ISP establish the link required to reference resources in any element attribute that allows references when using locators that would require creating a return channel link.

If the permission is not granted, the link will not be established and the link will fail as if the resource did not exist.

### 8.15.2.6 org.dvb.net.tuning.TunerPermission

This Permission is relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.2.6, "org.dvb.net.tuning.TunerPermission" on page 274.

Setting the location attribute of the Window object using locators that reference an audio or video stream will cause a SecurityException to be thrown.

Calling the open() Function attribute of the Window object using locators that reference an audio or video stream will cause a SecurityException to be thrown.

DVB-HTML applications can, if granted this permission, reference audio and video resources that require tuning using dvb: locators on the following element attributes:

- img.src
- input.src
- object.data
- Using the CSS uri() value

If the permission is not granted, no tuning will occur and the link will fail as if the resource did not exist.

Using a "dvb:" locator that references an audio or video resources on other element attributes is not required to be supported.

### 8.15.2.7 `javax.tv.service.selection.SelectPermission`

This Permission is relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.2.7, "`javax.tv.service.selection.SelectPermission`" on page 274.

Setting the location attribute of the Window object using locators that references a DVB service will cause a `SecurityException` to be thrown.

Calling the `open()` Function attribute of the Window object using locators that references a DVB service will cause a `SecurityException` to be thrown.

Applications can, if granted this permission, select away from the current service to a new service using dvb: locators that reference a DVB service on the following element attributes:

- `a.href`
- `area.href`

If the permission is not granted, no service selection will occur and the link will fail as if the resource did not exist.

Using a "dvb:" locator that references a DVB service on other element attributes is not required to be supported.

### 8.15.2.8 `org.dvb.user.UserPreferencePermission`

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.2.8, "`org.dvb.user.UserPreferencePermission`" on page 274.

### 8.15.2.9 `java.net.SocketPermission`

This Permission is relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.2.9, "`java.net.SocketPermission`" on page 274.

If the permission is granted (and if necessary the `rc` permission), the location attribute of the Window object may be set to a locator that references a resource on the host named in the permission file. If the permission is not granted then attempting to set this attribute with such a locator will cause a `SecurityException` to be thrown.

If the permission is granted (and if necessary the `rc` permission), the `open()` Function attribute of the Window object may be called with locator that references a resource on the host named in the permission file. If the permission is not granted then calling `open()` with such a locator will cause a `SecurityException` to be thrown.

DVB-HTML applications can, if granted this permission (and if necessary the `rc` permission), reference resources in any element attribute that allows such references using locators to resources on the host named in the permission file.

If the permission is not granted the link will fail as if the resource did not exist.

### 8.15.2.10 `org.dvb.media.DripFeedPermission`

This Permission is relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.2.10, "`org.dvb.media.DripFeedPermission`" on page 274.

Setting the location attribute of the Window object using locators that reference a dripfeed resource will cause a `SecurityException` to be thrown.

Calling the `open()` Function attribute of the Window object using locators that reference dripfeed resources will cause a `SecurityException` to be thrown.

DVB-HTML applications can, if granted this permission, reference dripfeed resources using dripfeed: locators on the following element attributes:

- img.src
- input.src
- object.data
- Using the CSS uri() value

If the permission is not granted, the link will fail as if the resource did not exist.

Using a "dripfeed:" locator on other element attributes is not required to be supported.

### 8.15.2.11 org.dvb.security.PrivilegedRCEPermission

When the permission request file requests the permission to do privileged runtime code extension on internal strings and this is granted, a PrivilegedRCEPermission with the name "internal" is created.

When the permission request file requests the permission to do privileged runtime code extension on external strings, but it is granted only for internal strings, a PrivilegedRCEPermission with the name "internal" is created.

When the permission request file requests the permission to do privileged runtime code extension on external strings and this is granted, a PrivilegedRCEPermission with the name "external" is created.

For behaviour, see "Use of strings in RCEs".

See 11.10.2.12, "org.dvb.smartcard.SmartCardPermission" on page 274

### 8.15.2.12 org.dvb.application.storage.ApplicationStoragePermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.2.11, "org.dvb.application.storage.ApplicationStoragePermission" on page 274.

### 8.15.2.13 org.dvb.smartcard.SmartCardPermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see 11.10.2.12, "org.dvb.smartcard.SmartCardPermission" on page 274.

## 8.16 Miscellaneous

### 8.16.1 Date Values

#### 8.16.1.1 Syntax

Absolute Date values have the following syntax:

### 8.16.2 Clock values

#### 8.16.2.1 Syntax

Absolute Clock values have the following syntax:

```
Clock-val          ::= ( Full-clock-val | Partial-clock-val | Timecount-val )
Full-clock-val     ::= Hours ":" Minutes ":" Seconds ( "." Fraction )?
Partial-clock-val  ::= Minutes ":" Seconds ( "." Fraction )?
Timecount-val      ::= Timecount ( "." Fraction )? (Metric)?
Metric             ::= "h" | "min" | "s" | "ms"
Hours              ::= DIGIT+; any positive number
Minutes            ::= 2DIGIT; range from 00 to 59
Seconds            ::= 2DIGIT; range from 00 to 59
Fraction           ::= DIGIT+
Timecount          ::= DIGIT+
2DIGIT            ::= DIGIT DIGIT
```

DIGIT ::= [0-9]

The default metric suffix is "s" (for seconds). No embedded white space is allowed in clock values, although leading and trailing white space characters will be ignored.

The following are examples of legal clock values:

Full clock values:

02:30:03 = 2 hours, 30 minutes and 3 seconds  
50:00:10.25 = 50 hours, 10 seconds and 250 milliseconds

Partial clock value:

02:33 = 2 minutes and 33 seconds  
00:10.5 = 10.5 seconds = 10 seconds and 500 milliseconds

Timecount values:

3.2h = 3.2 hours = 3 hours and 12 minutes  
45min = 45 minutes  
30s = 30 seconds  
5ms = 5 milliseconds  
12.467 = 12 seconds and 467 milliseconds

Fractional values are just (base 10) floating point definitions of seconds. The number of digits allowed is unlimited (although actual precision may vary among implementations).

For example:

00.5s = 500 milliseconds  
00:00.005 = 5 milliseconds

### 8.16.2.2 Offset values

Offset values are used to specify when an element should begin or end relative to another time.

An offset value has the following syntax:

offset-value ::= "+" ( Clock-value )

An offset value includes a sign on a clock value, and is used to indicate a positive offset. The offset is measured from the implicit start time:

### 8.16.3 Unrealisable locators

Locators may not be realisable for several reasons:

- The locator requires tuning for which permission or resources are not available
- The locator requires network or interaction channel access for which permission is not available
- The locator requires access to data that is denied by Conditional Access mechanisms
- The locator requires access to a authenticated file for which authentication fails
- The locator requires mounting a file system that cannot be mounted at the same time as the currently mounted file systems

#### 8.16.3.0.1 Presentation of Locators in DVB HTML.

It is implementation specific how the user agent indicates the presence of links to resources when rendering DVB-HTML. Where the user agent can determine a priori that a given link will fail, it is implementation dependent whether the user agent renders this as a link. It is also implementation dependent if and how a user agent presents failure to locate a resource due to insufficient permission being granted.



NOTE: [HTML 4 \[90\]](#) section 12.2.4 recommends that the user agent alerts the user when a referenced file is not available. To avoid computer-like dialogs and accessibility issues it is recommended that authors follow the guidelines in [Accessibility \[L\]](#) e.g. providing text equivalents using the `alt` attribute.

## 8.16.4 Relation to HTTP and HTTPS

The user agent string (see 8.13.1, "User agent strings" on page 163) is sent in a header as part of all HTTP and HTTPS requests (see 6.3.7, "Hypertext Transfer Protocol (HTTP)" on page 58).

## 8.16.5 DVB-HTML specific locators

DVB-HTML has two specific formats of locators which have defined semantics only in the context of DVB-HTML:

- The extended form of the DVB locator (see 8.16.5.1, "Extended DVB locator" on page 181)
- The exit locator used for application self termination (see 8.16.5.2, "Exit locator" on page 182)

### 8.16.5.1 Extended DVB locator

#### 8.16.5.1.1 Extended DVB locator syntax

The formal specification of the URL form expressed in BNF is given in the following extension to the `dvb:` locators defined in table 134, "DVB URL syntax" on page 353.

**Table 67 : Extended DVB URL syntax**

<code>dvbhtml_url</code>	= <code>dvb_url</code>   <code>dvb_scheme</code> ":" <code>dvbhtml_hierpart</code>
<code>dvbhtml_hierpart</code>	= <code>dvbhtml_net_path</code>
<code>dvbhtml_net_path</code>	= <code>"/"</code> <code>dvbhtml_entity</code>
<code>dvbhtml_entity</code>	= <code>dvb_service_contextual</code>   <code>dvb_service_component_contextual</code>   <code>ait_specifier</code>
<code>dvb_service_contextual</code>	= <code>"current"</code>   <code>"original"</code>
<code>dvb_service_component_contextual</code>	= <code>"current.audio"</code>   <code>"current.video"</code>   <code>"current.av"</code>
<code>ait_specifier</code>	= <code>ait_filter</code> "." <code>ait</code> <code>ait_abs_path</code>
<code>ait_filter</code>	= <code>"current"</code>
<code>ait_abs_path</code>	= <code>"/"</code> <code>ait_entity</code>
<code>ait_entity</code>	= <code>ait_root_directory</code>   <code>ait_application</code>
<code>ait_root_directory</code>	= <code>"app_root"</code>
<code>ait_application</code>	= <code>org_id_part</code> "." <code>app_id_part</code> [ <code>"?"</code> <code>ait_params</code> ]
<code>ait_params</code>	= <code>"arg_"</code> <code>1*digit</code> <code>"="</code> <code>*uric</code> [ <code>"&amp;"</code> <code>ait_params</code> ]

(for `digit` & `uric` see [RFC 2396 \[41\]](#))

(for `org_id_part` and `app_id_part` see 14.5, "Text encoding of application identifiers" on page 357)

The semantics of launching applications via such a locator are specified in 8.5.3.1.3, "Hypertext" on page 78.

#### 8.16.5.1.2 TV locators

A locator for a DVB service or service component can be a full `dvb:` locator, as defined in 14.1, "Namespace mapping (DVB Locator)" on page 353, or one of the following specific forms.

**Table 68 : (Sheet 1 of 2)**

Locator	Meaning
<code>dvb://current</code>	The service currently selected by the application.
<code>dvb://current.av</code>	The Audio and Video being presented on the background video device (see 11.6.2)
<code>dvb://current.audio</code>	The Audio being presented in association with the background video device
<code>dvb://current.video</code>	The Video being presented on the background video device

**Table 68 : (Sheet 2 of 2)**

Locator	Meaning
dvb://original	Originating service for this application (place of birth)
NOTE 1: Content authors who, in other systems, use the "tv:" locator, as defined in <a href="#">RFC 2838 [K]</a> , may use the equivalent "dvb://current.av" locator to reference the default audio and video component within the service.	

#### 8.16.5.1.3 Application locator

A locator for an application in the current service can be identified by the following specific forms Only applications that are visible in the application database using the current service filter can be found by this locator.

Selecting this locator will launch the application, with the associated parameters.

**Table 69 :**

Locator	Meaning
dvb://current.ait/Orgid.Appid?param=val&	An application in the service currently selected by the application.

#### 8.16.5.1.4 AIT locators

Locators for the root directory or the icon representation of the current application can be identified by the following specific forms.

**Table 70 :**

Locator	Meaning
dvb://current.ait/app_root	The root directory path as found in the dvb html application location descriptor for the application. (note 1)
dvb://current.ait/app_icon	The Icon found in the application icons descriptor for the application. (note 1)
NOTE 1: Shall mount file systems if referenced objects are in unmounted file systems and the link is followed.	

#### 8.16.5.2 Exit locator

In the context of a DVB-HTML application, actioning a link in the defined element, attribute context (see table 15, "MIME media type usage" on page 85) with the following form of locator shall cause an application to terminate.

exit:

The formal specification of the URL is given in the following BNF.

**Table 71 : Exit URL syntax**

exit_url	= exit_scheme ":" *uric
exit_scheme	= "exit"

Activating such a link shall request that the application manager move the current application into the **Killed** state. Any possible characters following the ":" shall be ignored in this version of the specification (see [RFC 2396 \[41\]](#)).

### 8.16.6 Domain

One of:

- The domain name of the server that served the document if it can be identified.
- An empty string ("" ) for a page with a dvb: locator.

NOTE: Such pages cannot therefore be in the same domain as any page delivered via http.

- Null otherwise.

## 9 Application model

### 9.1 Broadcast MHP applications

#### 9.1.1 Basic lifecycle control

The basic control of the lifecycle of broadcast MHP applications is through the selection of broadcast services. Selection of a broadcast service can be initiated by the user of the MHP terminal using the Navigator as well as by MHP applications offering EPG functionality.

The unit for the presentation and execution of content in the MHP specification is the service. A service in MHP represents a group of pieces of content which are intended to be presented together to the end-user. A service may consist of the contents of a broadcast DVB service, including audio/video streams, data streams and all the service information, applications and application signalling that is being broadcast. Other forms of service are possible, such as a stored service (see 9.7.2, "Stored services" on page 204).

Every service that gets presented by an MHP platform is presented within a service context. These form one of the foundations for the runtime environment and the execution model. A service context is an "environment" in which a service gets presented. It defines the boundaries of the service (letting the platform and applications identify which of the pieces of content that are being presented make up a given service). It also enables that service to be addressed and controlled as a single entity.

- In a DVB-J application, a service context is represented by an instance of the `ServiceContext` class. Where multiple DVB-J applications are being presented in the same service context, the number of `ServiceContext` objects representing a service context is implementation dependant, but each application sees only one such instance. Changes made by one application to the `ServiceContext` object that it has are visible to the `ServiceContext` objects representing the same service context in other applications. DVB-J applications may obtain a reference to the service context within which they are executing through using the method `getServiceContext (XletContext)` on the `ServiceContextFactory` class.
- In a DVB-HTML application, a service context is represented by a `JavaObject` object returned by the Bridge API, having the methods and attributes corresponding to the `ServiceContext` class. Where multiple DVB-HTML applications are being presented in the same service context, the number of `JavaObject` objects representing a service context is implementation dependant, but each application sees only one such instance. Changes made by one application to the `JavaObject` object that it has are visible to `JavaObject` objects representing the same service context in other applications. DVB-HTML applications may obtain a reference to the service context within which they are executing through the Bridge API to access the `getServiceContext (XletContext)` on the `ServiceContextFactory` class.

The means by which the navigator of an MHP terminal supports selection of services is implementation dependent. However, where an MHP application is using the numeric keys of the remote control, the navigator shall not respond to the user pressing these keys by causing service selection. Hence the user pressing the numeric keys to enter his pincode does not cause service selection.

A service context can present only one service at any one time. Selecting a service to be presented in a service context causes any previous service being presented in that service context to stop being presented. Any content part of the previous service which is not part of the new service shall stop being presented. MHP terminals may limit on the number of broadcast service contexts which can be presented simultaneously.

- In a DVB-J application, selecting a service corresponds to calling the `select()` method on such an instance.
- In a DVB-HTML, selecting a service either corresponds to activating a link to a `dvb: locator` representing a service, or by calling the `select()` method on a the `JavaObject` representing a service context.

## 9.1.2 Starting applications

When a broadcast service is selected, applications which are listed in the AIT of the service and identified as auto-start shall be launched as described in section 10.6, "Control of application life cycle" on page 215 without explicit intervention of the user. Applications which are started after the selection of a service will be controlled by signalling associated with that service. The MHP terminal shall monitor that signalling for changes made by the broadcaster. These changes may include the termination of particular applications as well as the addition of new auto-start applications.

Applications which are not identified as auto-start in the AIT shall not be automatically launched by the MHP terminal, but require explicit launching. This explicit launching can be done by the resident Navigator on the MHP terminal or by an MHP application. For example, the user can launch such applications after they have been offered a choice of applications through some user interface.

Where the currently selected service in a service context includes multiple MHP applications, any running applications may be able to launch other applications from that set. The launched applications shall be presented inside that same service context.

- A DVB-J application is able to achieve this using the application listing and launching API.
- A DVB-HTML application is able to achieve this either by activating a link using a locator referencing an application, or by using the Bridge APIs.

## 9.1.3 Support for execution of multiple simultaneous applications

The set of applications that are signalled within a service can be presented and executed concurrently.

MHP terminals shall be able to support applications from that set (using the same screen) at least as defined in the minimum platform capabilities section of this specification. MHP terminals are required to support execution of the set of such applications for each broadcast service which they permit to be presented simultaneously.

Broadcasters should ensure that simultaneous running of the set of applications for a service is comprehensible to the user and does not yield perceptible interference problems.

## 9.1.4 Stopping applications

MHP applications may stop themselves voluntarily using the MHP APIs or may be stopped by the MHP terminal in a number of situations. Examples of situations where this shall be allowed include:

### 9.1.4.1 A new service being selected replacing a previously selected one

When a new service is selected and replaces a previously selected one, applications from the former service shall only continue to execute where they are signalled in the new service. If an application is not signalled in that signalling then it will be stopped by the MHP terminal. Where an application is known to be bound to a single service, the broadcaster can identify that application as service bound using the `service_bound_flag` in the application descriptor. Such applications shall be stopped as soon as possible by the MHP terminal and without needing the AIT for the new service to be available. This allows the autostart application(s) of the new service to be started earlier than would otherwise be the case.

### 9.1.4.2 The stopping of an application by another application

Subject to the security policy of an MHP terminal, one application may request to stop another application. In such a case, the resident Application Manager, after a successful security check, kills the application otherwise that application shall continue running, without interruption.

### 9.1.4.3 Changes in the application signalling to request a particular application be stopped

The broadcaster may request an MHP terminal to stop an application using the control codes in the AIT. The precise semantics of these are dependent on the application format.

#### 9.1.4.4 Stopping by the MHP terminal due to a shortage of resources

Where an MHP terminal has insufficient resources to continue the execution of one of the running applications, the MHP terminal is allowed to decide to stop an application without user intervention (e.g. memory, CPU and resources managed by the resource management API). When this happens the MHP shall first kill the applications by increasing order of the priority fields signalled in the application\_priority field in the application descriptor for each application.

NOTE: The precise resources of an MHP terminal are implementation dependant.

#### 9.1.5 Persistence of Applications Across Service Boundaries

Where a running application is signalled in both the new service and the former service, and is not signalled as service bound in the former service, it shall continue to run and shall not be restarted. In this case, the running application shall become controlled by the application signalling of the new service where it is signalled and not the signalling of the former service. Hence the MHP terminal shall monitor the AIT of the new service and shall stop responding to the AIT of the former service.

If the application is signalled as service bound in the former service then it is terminated in the normal way as the new service is selected. If it is signalled as auto-start in the new service it will restart with no volatile context from the previous instantiation.

If an application survives a service selection operation it shall run in the new service context without automatically getting any access to the broadcast file system of the new service. The broadcast file system shall be lost as defined in 6.2.5.3, "Loss of Carousel Behaviour" on page 56. Applications are responsible for implementing their own strategy for continuing to execute after this point.

#### 9.1.6 Management of autostarting

The receiver shall launch autostart applications under the following conditions:

- the signalling indicates that the application can be supported by the receiver, as defined by the application profile & versioning information contained in the application descriptor,
- only a single application with a given [Application identification](#) is allowed to run at any time,
- the application is a newly introduced autostart application or has newly been given autostart status.

So:

- when a service is selected the receiver shall launch at most one instance of each autostart application that it can support,
- if after service selection an autostart application that the receiver can support is introduced or a previously listed supportable application gains autostart status then the application shall be launched subject to normal resource limitations, etc.

However, if an autostart application terminates it is not restarted unless it again becomes a new autostart application. An application becomes a new autostart application in the following cases:

- The receiver navigates away from the service and then selects a service where the application is autostart.
- The application is removed from the AIT and then is re-introduced.
- The autostart status of the application is reset then set again.

NOTE 1: In summary the autostart status of an application is in effect an edge trigger rather than level trigger signal.

NOTE 2: These semantics for the autostart behaviour address "de-bouncing" the case where an autostart application terminates voluntarily. They do not address the case where the receiver terminates the application.

NOTE 3: This specification does not describe in detail the timing required for the broadcast signalling to renew the autostart status of an application.

### 9.1.7 When tuning is not service selection!

MHP applications may cause tuning to another transport stream by mechanisms other than service selection. Usage of these mechanisms does not constitute service selection and therefore no applications from the target transport stream or service shall be started either by the MHP terminal or by MHP applications. The MHP terminal shall continue monitoring the AIT of the logically selected service where this is available on the target transport stream. Where the AIT of the selected service is not available, the application shall continue executing as described in 10.4.4, "Visibility of AIT" on page 211. The service being presented in the service context shall not be changed by an application using these mechanisms:

- DVB-J tuning API or streamed media APIs
- DVB-HTML accessing a media or service reference (e.g in an <object> or <img> element), or by accessing the tuning API through the Bridge

### 9.1.8 MHP Applications and Service Selection

MHP applications may select services using the service selection API. The service selection API includes a class `ServiceContext` to represent environments in which services may be presented. Calling the select method on a `ServiceContext` causes a new service to be presented in that context and any former service being presented in that context will be stopped.

Where one MHP application uses the application listing and launching API to successfully start a second MHP application, the second MHP application shall be considered as executing inside the service context of the first MHP application. The number of `ServiceContext` objects representing a service context is implementation dependant, but each application sees only one such instance. Changes made by one application to the `ServiceContext` object that it has are visible to the `ServiceContext` objects representing the same service context in other applications.

MHP applications started in response to a service selection operation are considered to be executing "inside" a service context. They may obtain a reference to the service context within which they are executing through using the method `getServiceContext(XletContext)` on `javax.tv.service.selection.ServiceContextFactory`.

MHP applications may cause tuning to another transport stream directly using the tuning API or, in the case of DVB-HTML, indirectly by reference to a media stream. Such tuning does not constitute service selection and therefore no applications from the target transport stream or service shall be started either by the MHP terminal or by MHP applications. The MHP terminal shall continue monitoring the AIT of the selected service where this is available on the target transport stream. Where the AIT of the selected service is not available, the application shall continue executing as described in 10.4.4, "Visibility of AIT" on page 211. The service being presented in the service context shall not be changed by usage of these APIs.

### 9.1.9 Broadcast service related stored applications

These are applications designed to operate in the context of one or more broadcast services. As a consequence even when stored at the receiver they are not able to run unless suitably signalled in the AIT of a currently selected broadcast service. This includes all applications listed in the AIT whether launchable or externally authorised. These applications include the following:

- applications which have been proactively cached by the MHP terminal implementation
- applications which have been explicitly stored as part of the current selected broadcast service
- applications which have been explicitly stored and form part of a stored service and which happen to also be present in the signalling of the currently selected broadcast service

This class of application may have technical dependencies which mean that they cannot be operate correctly unless the MHP terminal is also decoding one of the required broadcast services. For example, an EPG application might be stored to enable it to load more quickly once selected but requires access to the broadcast stream to provide it with data to present.

Stored broadcast related applications shall take the lifecycle model of broadcast applications. E.g.

- following a service change the application behaviour will depend on the AIT signalling in the newly selected broadcast service.
- the application can only be started when it is listed in the AIT of the currently selected service.

In summary, the behaviour is that of a broadcast application, except that the loading is possibly accelerated due to the file system cache provided by the storage.

When one of these applications executes, it shall see the same filesystem as if it was being run directly from broadcast. The set of files stored in response to being listed in the application description file (see 9.7.1, "Applications loaded from the interaction channel" on page 204) shall always over-ride the same files from the broadcast.

NOTE: The application provider is responsible for managing any dependencies between those files which can be stored and those files which are not required to be stored.

### 9.1.9.1 Version management

If the cached/stored and currently signalled version numbers do not match, the MHP terminal shall start the broadcast version. If the application has `not_launchable_from_broadcast` set to '1' then it shall not be launched at all.

If the version number of a broadcast application is newer than that of the corresponding stored application it is the responsibility of the newer broadcast application to manage the update. Storing the newer version will naturally overwrite the previously stored version. If `not_launchable_from_broadcast` is set to '1' then a launchable application must be provided to manage the update (although an MHP terminal may perform the update autonomously).

NOTE: MHP terminals may refuse to update, or keep stored, applications where the version numbers change too frequently. Broadcasters should ensure that frequently changing files are not listed in the "Application description file".

## 9.2 DVB-J Model

### 9.2.1 Starting DVB-J Applications

DVB-J applications may be started by any of the means defined for general MHP applications. The application listing and launching API defined in annex S, "(normative): Application Listing and Launching" on page 711 allows one MHP application to start another MHP application subject to security policy. The `start()` method of the `AppProxy` interface will then cause the Application Manager to start the new MHP application subject to normal resource limitations.

The Xlet interface is defined in the `javax.tv.xlet.Xlet` interface [Java TV \[51\]](#). DVB-J applications provide a class implementing this interface and reference that class in the DVB-J application location descriptor. In order to start a DVB-J application, the application manager shall call the constructor of this class, the `initXlet()` and the `startXlet()` methods of this interface.

### 9.2.2 Stopping a DVB-J Application

DVB-J applications may stop for any of the reasons listed for general MHP applications. An application shall be able to notify that it is stopped by finishing its execution and informing the Application Manager through the `notifyDestroyed()` method on the `javax.tv.xlet.XletContext` interface. This interface also includes other methods to allow a DVB-J application to request or notify changes in its state.

The application listing and launching API allows an application to indirectly control the lifecycle of another application subject to security policy. This control is indirect because an application cannot invoke an Xlet state method directly, but goes through this API. This ensures that the resident Application Manager can always keep track of all the application that are running.



When a DVB-J application is stopped by an MHP terminal, the `destroyXlet` method of the signalled Java class implementing the Xlet interface, i.e. the start of the application shall be called by the application manager. In the case of the application being stopped due to a service selection operation, the stopping of the application shall be unconditional. This method call gives applications their last opportunity to save state before their execution stops. Applications which wish to survive the user of the MHP terminal zapping away from their service (e.g. during an advertising break) must save their state and reload that state when they are re-started if the user returns to that service later.

## 9.2.3 DVB-J Application Lifecycle

### 9.2.3.1 Introduction

This section describes the Xlet lifecycle model for the DVB-J API. This describes the capabilities of the Xlet in each state and the methods by which the application manager influences the life cycle state. This section is not directly related to other aspects of a system, such as graphics or shared resource allocation/management.

**NOTE:** The traditional Java platforms define a number of application models that have their own lifecycles associated with them. In general, they are designed to address specific issues on that platform. For instance, the Applet was designed to provide support for executable content in web pages. However, none of the existing application technology fully addresses the specific requirements of television receivers. The application lifecycle defined in this section is meant to be compatible with existing Java platforms and virtual machine technology.

### 9.2.3.2 DVB-J Application Lifecycle State Machine

The Xlet state machine ensures that the behaviour of an Xlet is close to the behaviour that television viewers expect, specifically:

- The perceived start-up latency of an Xlet can be very short.
- It is possible to put an Xlet into a state where it is not providing its service.
- It is possible to destroy an Xlet at any time.

The figure 15 shows the state machine model for Xlets. The Xlet states are defined in more detail in table 72.

The different influences that can cause an Xlet to change state include:

- The application manager uses the [Xlet API](#) to signal these changes to the Xlet.

Various factors may stimulate the application manager to act in this way, for example:

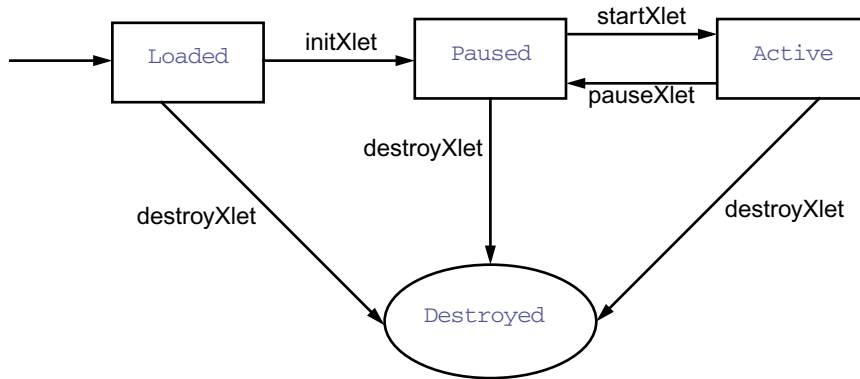
- Broadcast signalling (e.g. a change in the state of the `application_control_code` parameter carried by the AIT (see 10.4, "Application Information Table" on page 210)).
- User selection of an application in a host provided UI



- The Xlet itself "decides" to change state

The application uses the `XletContext` Object to communicate or request such changes to the application manager.

- Another Xlet acts via the application launching API (see 11.7.2, "Application discovery and launching APIs" on page 261).



**Figure 15 : Xlet lifecycle state machine diagram**

Table 72 : Valid DVB-J application lifecycle states (Sheet 1 of 2)

State Name	Description
Loaded	<p>The DVB-J application has been loaded and has not been initialized.</p> <p>The signalled Java class used to initiate a DVB-J application must implement the <code>javax.tv.xlet.Xlet</code> interface. Otherwise, the class (and hence the application) may be ignored. An instance of the signalled Java class is created by the application manager e.g. using the <code>Class.newInstance</code> method. Therefore a DVB-J application must have a public "default constructor". Otherwise, the class (and hence the application) will be ignored. If the default constructor returns without throwing an uncaught exception, then the application is considered to be "Loaded", otherwise the application immediately enters the <code>Destroyed</code> state and is discarded. Once the application has been successfully loaded and instantiated, the application manager can transition the application to the <code>Paused</code> state by invoking the <code>initXlet</code> method on the signalled class (implementing the <code>Xlet</code> interface).</p> <p>If the <code>initXlet()</code> method throws an <code>XletStateChangeException</code> then the application shall remain in the Loaded state. The only possible state transition for such an application is into the destroyed state. The application can request this itself or wait for the application manager to cause this transition.</p> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• Application initialisation is intended to occur in the <code>initXlet</code> method, rather than in the default constructor.</li> <li>• This state is entered only once per instance of an DVB-J application.</li> </ul>
Paused	<p>A <code>Paused</code> DVB-J application should minimize its usage of resources if it wants to maximize its probability of survival. This does not imply that it cannot be holding any resources, but in such a case, it would have a lower priority as concerns access to resources than it had when it was in the <code>Active</code> state.</p> <p>This state is entered:</p> <ul style="list-style-type: none"> <li>• From the <code>Loaded</code> state after the <code>Xlet.initXlet()</code> method returns successfully when invoked by the application manager (the first time). Other invocations of this method do not cause the change of state.</li> </ul> <p>Note that the application manager shall only call <code>initXlet()</code> once per instance of a DVB-J application.</p> <ul style="list-style-type: none"> <li>• From the <code>Active</code> state after the <code>Xlet.pauseXlet()</code> method returns successfully when invoked by the application manager. Other invocations of this method do not cause the change of state.</li> <li>• From the <code>Active</code> state upon entering the <code>XletContext.notifyPaused()</code> method.</li> </ul>

**Table 72 : Valid DVB-J application lifecycle states (Sheet 2 of 2)**

State Name	Description
Active	<p>The DVB-J application is functioning normally and providing service.</p> <p>This state is entered:</p> <ul style="list-style-type: none"> <li>From the <code>Paused</code> state after the <code>Xlet.startXlet()</code> method returns successfully.</li> </ul>
Destroyed	<p>The DVB-J application has released all of its resources and terminated.</p> <p>This state is entered:</p> <ul style="list-style-type: none"> <li>When the Xlet's <code>destroyXlet()</code> method returns successfully. The <code>destroyXlet()</code> method shall release all resources held and perform any necessary clean up so it may be garbage collected.</li> <li>Upon entering the <code>XletContext.notifyDestroyed()</code> method. The Xlet performs its clean up actions before calling the <code>notifyDestroyed()</code> method.</li> </ul> <p><b>Note:</b> This state is only entered once.</p>

Only the DVB-J application can determine if it is able to provide the service for which it was designed. As such, in some respects an application manager cannot guarantee whether an DVB-J application can, or is, providing its service; and application manager can only indicate that the DVB-J application is able to do so. A typical sequence of DVB-J application execution is:

**Table 73 : Typical DVB-J application lifetime walk through**

Application Manager	DVB-J application
The application manager creates a new instance of an Xlet.	The Xlet's default (no argument) constructor is called, it is in the <code>Loaded</code> state.
The application manager creates the necessary context object for the DVB-J application to run, and initializes the Xlet.	The DVB-J application uses the context object to initialize itself. It is now in the <code>Paused</code> state.
The application manager has decided that it is an appropriate time for the DVB-J application to perform its service, so it signals it to enter the <code>Active</code> state.	The DVB-J application acquires any resources it needs and begins to perform its service.
The application manager no longer needs the DVB-J application to perform its service, so it signals the DVB-J application to stop performing its service.	The DVB-J application stops performing its service and might choose to release some resources it currently holds.
The application manager has determined that the DVB-J application is no longer needed, or perhaps needs to make room for a higher priority application in memory, so it signals the DVB-J application that it is a candidate to be destroyed.	If it has been designed to do so, the DVB-J application saves state or user preferences and performs clean up.

## 9.2.4 Xlet API

The Xlet API provides MHP application developers with an API that provides life cycle signalling. The Xlet API uses the callback approach to signal state changes.

This API is specified in section 11.7.1, "APIs to support DVB-J application lifecycle" on page 261.

### 9.2.4.1 Xlet State Change Semantics

An Xlet's state can change either by having one of the methods on its Xlet Interface called, or by making an internal state transition and notifying the application manager via the `XletContext` Object. The semantics of when that state change actually happens are important:

- Calls to `Xlet`: this interface indicates a successful state change only when the call successfully returns.
- Calls to `XletContext`: the `notifyDestroyed()` and `notifyPaused()` methods indicate a state change on entry. The `resumeRequest()` method indicates no state change, instead only a request to change state.

If a method on the Xlet interface throws an `XletStateChangeException`, the Xlet shall remain in the state it was in immediately prior to the call of the method throwing the exception unless otherwise specified. In this specification, the only exception to this rule is the `destroyXlet` method when the `unconditional` flag is `true` where throwing the `XletStateChangeException` is specified to have no effect. For the case of `initXlet` which may only be called once, the application manager may choose to transition the Xlet to the destroyed state (without calling `destroyXlet`) some implementation specific time later.

### 9.2.4.2 Xlet state change requests

The following table defines those states in which calls to the methods on `XletContext` relating to state management are valid;

**Table 74 : States for valid state management calls**

Call	State
<code>notifyDestroyed</code>	all states
<code>notifyPaused</code>	started only
<code>resumeRequest</code>	paused only

Calls to these methods when an Xlet is in any other state shall have no effect.

## 9.2.5 Multiple application environment support

The DVB-J platform allows for the simultaneous execution of several DVB-J applications.

Allowing several DVB-J applications to run simultaneously implies that some rules be defined for these DVB-J applications to share the resources of the MHP, and in particular for them to share the Input Focus and the Output Focus.

### 9.2.5.1 Control of DVB-J applications by other DVB-J applications

The MHP provides support for control of the lifetime of a DVB-J application by another DVB-J application. This feature enables broadcasters to write their own "Launcher applications" that take care of the presentation to the user of the availability of DVB-J applications, and that enables eventually the user to launch DVB-J applications. Note that the actual control of the lifetime of an DVB-J applications is done by the Application Manager only. The MHP only provides APIs that enable DVB-J applications to ask the Application Manager to start, stop, pause and resume DVB-J applications.

See [11.7.2, "Application discovery and launching APIs" on page 261](#).

### 9.2.5.2 Input Focus management

The input focus is defined as follows:

- the application that has input focus is in principle able to receive user-input events.
- other applications not having the input focus can request to receive a subset of user-input events via a dedicated API. See ["org.dvb.event" on page 246](#).

### 9.2.5.3 Other resources management

The APIs defined in this specification provide support for resource allocation/revocation and resource revocation notification. The semantics of the APIs, however does not define under which circumstances an access to a resource is granted or revoked. While it is well understood that in most cases, it is up to the MHP implementation to define its own policy in terms of resource management, this section defines the basic rules that an MHP implementation has to follow.

The MHP specification describes a multi-application environment. Hence several applications may be competing for access to the same atomic resource. The resource notification API described in section 11.7.5 on page 266 provides a common way for applications to negotiate access to scarce atomic resources when such competition happens. This API allows for the MHP terminal to inform the application that currently holds the resource that another application wants to access this resource. It also provides a means for the owner of the resource and to the requester of the resource to communicate by private means. This private communication is reflected by the request\_data object that the requester may pass to the owner. The semantics of this object is private and has to be known by both applications.

Some existing and general purposes java APIs that were developed before the MHP work was started do not use this general resource sharing mechanism. Hence access to resources addressed by these APIs are not subject to negotiation. For example, when an application holds a JMF player, if another application was to create a JMF player for the same content-type, the MHP has to decide by itself whether it withdraws the resource underlying the JMF player from the current owner and grants it to the requester.

It is also possible for applications to use the inter-application communication API to establish private communication channels enabling them to negotiate access to resources.

### 9.2.5.4 VM implementation

Where there are multiple DVB-J applications being executed as part of the same service, MHP terminals are allowed implement these in a single actual virtual machine instance. Regardless this shall conform to 11.2.1, "Basic Considerations" on page 241.

## 9.3 DVB-HTML Model

### 9.3.1 The DVB-HTML Application

#### 9.3.1.1 DVB-HTML Application

A DVB-HTML application is defined as a set of documents selected from the DVB-HTML family of elements and content formats as defined in the specification. The extent of the set is described by the application boundary.

#### 9.3.1.2 User agent

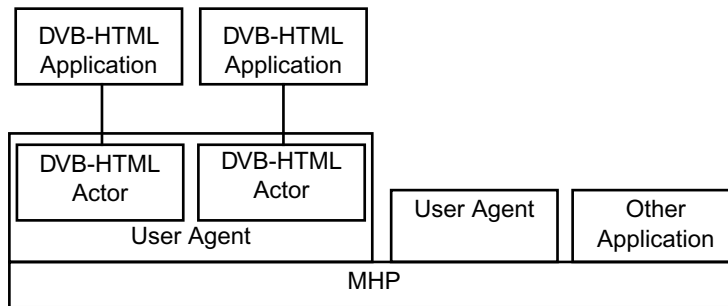
A user agent is an application that interprets a content format (in this case DVB-HTML documents).

Note This could be implemented as a plug-in.

#### 9.3.1.3 DVB-HTML Actor

A DVB-HTML actor is defined as the locus of activity or process involved in running the specific set of DVB-HTML documents for some DVB-HTML application, plus any instantiated context for that data. The actor runs inside a user agent (native, plug-in or downloaded). The nature of the process is not defined explicitly as it depends on the nature of the user agent itself. More than one such locus of activity may be present in any given user agent.

There is a single DVB-HTML Actor for each running DVB-HTML Application, each DVB-HTML Application can consist of multiple documents several of which could be simultaneously displayed.



**Figure 16 : Relationships between actors and applications**

### 9.3.1.4 Application boundary

The application boundary defines the namespace of the DVB-HTML application, all content documents within this namespace are considered to be part of the DVB-HTML application, content documents outside the namespace are considered to be in other DVB-HTML applications. This namespace is used to determine: to which application each piece of cached material belongs, to assist an MHP in more efficient pre-fetching and to facilitate storing applications.

The logical extent of a DVB-HTML application could potentially be quite large, and for various reasons might not all be on the MHP terminal at one time. Part of it might be broadcast, part of it might be on local storage, part of it might be on the world wide web - some of it may even be generated on demand. For this reason the compact format of the regular expression is used to define the extent. The set of DVB-HTML documents making up a DVB-HTML application is defined by a regular expression over the locator language, broadly a locator consists of a text string in the following form from RFC 2396 [41] and acts as the glue which holds the application together.

```
scheme://host/dir1/dirn/file#subref
```

A regular expression is the definition of a set by a pattern which can test whether a given string is or is not a member of the set, for example the regular expression:

```
https?://www\.(dvb|etsi)\.org/[a-z0-9/]+\\.html?
```

Matches the logical locator of any file on both www.dvb.org or www.etsi.org, either reached by http or https, if and only if it is a DVB-HTML file (its name ends in ".htm" or ".html"), and its pathname contains only alphanumeric characters. Quite terse definitions can match a large set of files [see for example Compilers [C]].

#### 9.3.1.4.1 Regular Expression Syntax

A regular expression (RE) specifies a set of character strings to match against. A member of this set of strings is said to be matched by the regular expression.

In order for a locator to match a boundary regular expression the whole locator must be matched by the whole regular expression; any parameters (characters including and after the first "?" or "#" in the locator) are not considered as part of the locator for purposes of boundary matching.

The form of regular expression used for defining application boundaries is defined as a POSIX Extended Regular Expression from POSIX [59] section 2.8.4.:

Relative locators in the DVB-HTML application are expanded to a full URI as defined in RFC 2396 [41], (the default base URI being that carried in the application location descriptor) before being matched.

A pattern may be broken into sub patterns in a set of application boundary descriptors (see signalling). The full pattern is formed from the OR of all the sub patterns. Each application boundary descriptor may be associated with a label (see 10.10.3, "DVB-HTML application boundary descriptor" on page 233). This label can be used for pre fetching in a transport specific manner, for example in an object carousel it defines that all modules matching the label should be preloaded.

For example: an application consists of an entry web page /phase0/index.html, and is factored into three sub sections, each of which has an associated stylesheet and image directory.

```
labelA: (/phase0/.\.html | /phase0/images1/.\.png | /phase0/scripts1/.\.js)
labelB: (/phase1/.\.html | /phase1/images/.\.png | /phase1/scripts/.\.js)
labelC: (/phase2/.\.html | /phase2/images/.\.png | /phase2/scripts/.\.js)
```

The entry point locator signalled for this application matches the first regular expression, this allows the pre-fetch mechanism to load the modules labelled with labelA (which the broadcaster arranges to contain the contents of directory phase0), Once the user agent is running, it can use this information to detect which if any links from the current page might transition to a new phase and therefore require more pre-fetching.

## 9.3.2 DVB-HTML Application Lifecycle

### 9.3.2.1 Introduction

There are three key parts of the DVB-HTML application lifecycle model:

- a) How applications are signalled as available to the MHP, and for auto start and prefetch applications how the start time is synchronized with any associated media stream.
- b) How and when the application manager or other launcher application makes the presence of an non auto-start application known to the user and provides it with a trigger. This is covered by the application discovery and launching mechanisms.
- c) How a broadcaster controls an actor after it has started.

### 9.3.2.2 Signalling

The DVB-HTML Application is signalled as described in chapter 10, "[Application Signalling](#)" on page 208.

The application manager can be requested to start a DVB-HTML application either because it is signalled as auto start, or through the application launching API.

On receiving the request that a DVB-HTML application is to be started (i.e. an [AUTOSTART](#) or [PREFETCH](#) appears in the AIT or it is user instantiated), and there is no application with the same applicationID already instantiated, the application manager should attempt to find a suitable user agent. It can also at this point begin pre-fetching material.

If the application manager is unable to instantiate a user agent either through lack of resources, or no suitable user agent being available then any pre-fetching can be aborted, and any trigger signal can be ignored.

It is platform dependant at what time a DVB-HTML autostart application starts. For a pre-fetched DVB-HTML application a trigger is required which carries the time at which the application should start providing service.

The DVB-HTML actor can be in one of 5 DVB-HTML Application states.

- [Loading](#)
- [Active](#)
- [Paused](#)
- [Destroyed](#)
- [Killed](#)

Each of these states has a precise meaning outlined in the following sections. The transitions between states are made as a result of, for example:

- A trigger, such as a request to go to a new document,
- A trigger such as the DVB-HTML application making an explicit request to change state,
- A change in the external environment i.e. the application\_control\_code in the AIT changes.

Since a user agent may be performing as several actors, it can be in several of these states at one time, each actor however will be labelled with a unique application ID.

A DVB-HTML application proceeds by moving between documents, while the documents remain within the DVB-HTML application boundary the DVB-HTML application continues to run normally.

Links within an DVB-HTML application normally replace the existing document, but attributes may be present on a link which cause both the new and old document to be visible at the same time.

### 9.3.2.3 Lifecycle control

The state model for the DVB-HTML application lifecycle control model described in this section reflects the signalling (see 10.6, "Control of application life cycle" on page 215) and is an abstract view of how a DVB-HTML application operates, and considers the kinds of resources that a user agent would need in order to function properly: resources concerned with output (rendering), input (event catching) and connection (the availability of the content).

The abstract model however is mostly illustrative and does not imply any resource management strategy nor is it intended to overly constrain the implementation of a user agent;

#### 9.3.2.3.1 State diagram

The following transition diagram summarizes the states and the transitions between them.

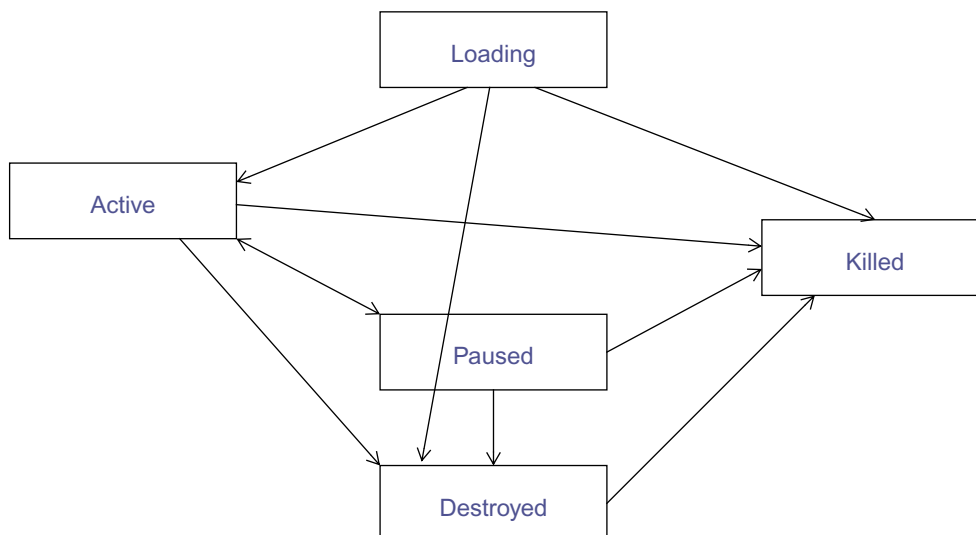


Figure 17 : DVB-HTML application life cycle state diagram

### 9.3.3 The State Model

The entry state of the state machine, Loading, is characterized by access to the content resources and signalling resources but does not have or require input and output resources. This implies the actor can prefetch content and receive triggering events for transition to the Active state, but will not be presented to the viewer.

On entry into the Active state the actor would be assumed to have full access to the content of the current document and all resources of the MHP, subject to resource management and security issues.

The paused state is a reduced operational state. If the application manager or the user agent needs resources for other purposes, an actor may be moved to the paused state, when in this state it may no longer have full (or even any) access to resources. When the actor is reactivated it returns to its previous state.

The destroyed state can be characterized as loss of the content resource. The actor may still be able to run the DVB-HTML application due to caching or other mechanisms but must be prepared for loading of some or all of the documents from within the DVB-HTML application to fail. It is implementation dependent how such failure is handled. This is a way for the broadcaster to signal to the MHP that it is on it's own.



The killed state is characterized by the loss of all resources, and is the signal for actions concerned with cleanup of the actor. The MHP reclaims whatever resources it deems necessary. It is implementation dependent whether cached material is disposed of.

If the AIT signal is **KILL**, an actor is forcibly terminated (and all resources associated with it reclaimed) regardless of state.

### 9.3.3.1 Loading

#### 9.3.3.1.1 Name

Loading

#### 9.3.3.1.2 Entry actions

Instantiation of an actor.

#### 9.3.3.1.3 Activities

Waiting for documents to be available and loading documents without rendering them.

#### 9.3.3.1.4 Resources

Content, signalling, Output

#### 9.3.3.1.5 Transitions

Active. preconditions:

- enough data is available to present something sensible.

Killed. preconditions:

- If the DVB-HTML application is signalled as **KILL**,

Destroyed. preconditions:

- If the DVB-HTML application is signalled as **DESTROY**,

#### 9.3.3.1.6 Comment

This is the entry state of the state machine This state is entered only once in the lifetime of the DVB-HTML actor. Any start-up phase of a user agent can also be considered as part of this state. When an actor is in this state it is not rendering anything. This state should not be confused with any prefetching of modules which may be carried out by the MHP prior to application launch.

### 9.3.3.2 Active

#### 9.3.3.2.1 Name

Active

#### 9.3.3.2.2 Activities

Gathering and parsing current document and related resources., rendering document, Maintaining rendered documents. receptive to events, waiting for triggering event to show loaded documents.

#### 9.3.3.2.3 Entry actions

If application is signalled as pre-fetch wait for trigger before displaying anything.

#### 9.3.3.2.4 Resources

Content, signalling, output, input.

### 9.3.3.2.5 Transitions

Pause.

- If the user agent or application manager puts the DVB-HTML application in PAUSE.

Destroyed.

- If the DVB-HTML application is signalled as DESTROY.

Killed.

- If the DVB-HTML application is signalled as KILL

### 9.3.3.2.6 Comment

This state is the steady state,

In this state it is user agent specific as to whether a partially loaded document is displayed, or deals with input triggers.

If a transition is made to a new document within the application, the actor remains in this state

if a related resource document of the main document changes, then the resource may be reloaded, causing the DVB-HTML actor to receive appropriate DOM events, however the DVB-HTML actor is not considered to change state. Similarly the DVB-HTML actor may gain and lose the focus while in this state, receiving the appropriate DOM events – it may receive fewer input events when it does not have the focus.

If the AIT no longer refers to the DVB-HTML application no special action is taken for DVB-HTML actors that are in the Active state.

## 9.3.3.3 Paused

### 9.3.3.3.1 Name

Paused

### 9.3.3.3.2 Activities

DVB-HTML actor should minimise its use of resources.

### 9.3.3.3.3 Resources

Application specific.

### 9.3.3.3.4 Transitions

Active.

- If the DVB-HTML application is resumed.

Destroyed.

- If the DVB-HTML application is signalled as DESTROY.

Killed.

- If the DVB-HTML application is signalled as KILL

### 9.3.3.3.5 Comment

The semantics of this state are both user agent and DVB-HTML application specific. When the DVB-HTML application returns from the "Pause" state, the environment might have changed (loss of resources or network connections) and some events may not have been reported.

### 9.3.3.4 Destroyed

#### 9.3.3.4.1 Name

Destroyed

#### 9.3.3.4.2 Activities

Loading documents. Rendering documents. Consuming events. Interact with the user.

#### 9.3.3.4.3 Resources

input and output.

#### 9.3.3.4.4 Transitions:

Killed.

- {If the DVB-HTML application is signalled as KILL} OR {local event forces the actor to terminate, possibly through application manager}

#### 9.3.3.4.5 Comment

This state indicates the MHP may no longer be able to access the content resources required to run the DVB-HTML application. It is DVB-HTML application and user agent specific as to whether the actor continues to run, and if it does how the user should be informed if any link is no longer available because the content it refers to is no longer available, or a cached copy has expired. The DVB-HTML actor may continue to execute in the destroyed state until the user actively dismisses it.

### 9.3.3.5 Killed

#### 9.3.3.5.1 Name

Killed

#### 9.3.3.5.2 Entry actions

Release of resources.

#### 9.3.3.5.3 Activities

Termination of the DVB-HTML application.

#### 9.3.3.5.4 Resources

none.

#### 9.3.3.5.5 Transitions

none

#### 9.3.3.5.6 Comment

After the activities in this state are finished, the application is deemed to have terminated. This state is the exit state of the state machine.

## 9.4 Application activity events

Each page within an application shall receive load and unload events with the following constraints:

- The DVB lifecycle `AppActive` event precedes the first DVB-HTML `load` event.
- There is no guarantee on the relative ordering of the first DVB-HTML `load` event and the DVB lifecycle `AppStarting` event.
- When a document is unloaded because of a transition to a new page the current document will receive an `unload` event and all event handlers will be allowed to complete before the new page begins its lifecycle.
- When an application transitions to the Killed state (i.e., the DVB-HTML actor ceases to execute), there is no guarantee that the current document will receive any `unload` event.
- When an application transitions to the Destroyed state the application may continue to execute in the Destroyed state, in which case `unload` events and new `load` events may be received if assets are available
- The document `load` event is received after all parts that can receive `load` events have received them. Specify when the `load` event is fired for each media type (e.g. streamed, multipart replace)
- No non-lifecycle events are delivered to a page until after the `DVBDOMStable` event is delivered
- Events that occurred when the application is in the from the Loading state and may be delivered when in the Active state

**NOTE** The first two constraints will not be visible to documents in frames which share an application boundary with their parent, as they do not receive DVB lifecycle events.

The following state diagram illustrates these interactions.

- DVB-HTML lifecycle states and transitions are in This Style.
- DOM events are in **This style**.
- States and transitions which are only part of this model are in *this style*
- The *request* transition indicates a request (from user inputs, trigger events etc.) to navigate to a new page.
- The *(no event)* transition may happen immediately, with no other input.
- The *loading page* state covers loading content for a new page and may cover full or incremental rendering of the page (although the page may not be presented to the user, if the application is signalled as PREFETCH and the AppStarting event has not been received).
- The *loading content* state covers the referenced content for a new page after the initial content is parsed and may cover full or incremental rendering of the page (although the page may not be presented to the user, if the application is signalled as PREFETCH and the AppStarting event has not been received).
- In the *loaded page* state, the page has been loaded. Unless the application is signalled as PREFETCH and no AppStarting event has occurred, it will also have begun to be presented to the user.
- In the *unloading page* state, content loading for the new page may begin.
- The AppStarting, AppPause, AppResume and AppDestroyed DVB-HTML lifecycle events are not shown in this diagram, because the only constraints on them are in relation to the standard MHP lifecycle states and events, not in relation to the DOM load and unload events.

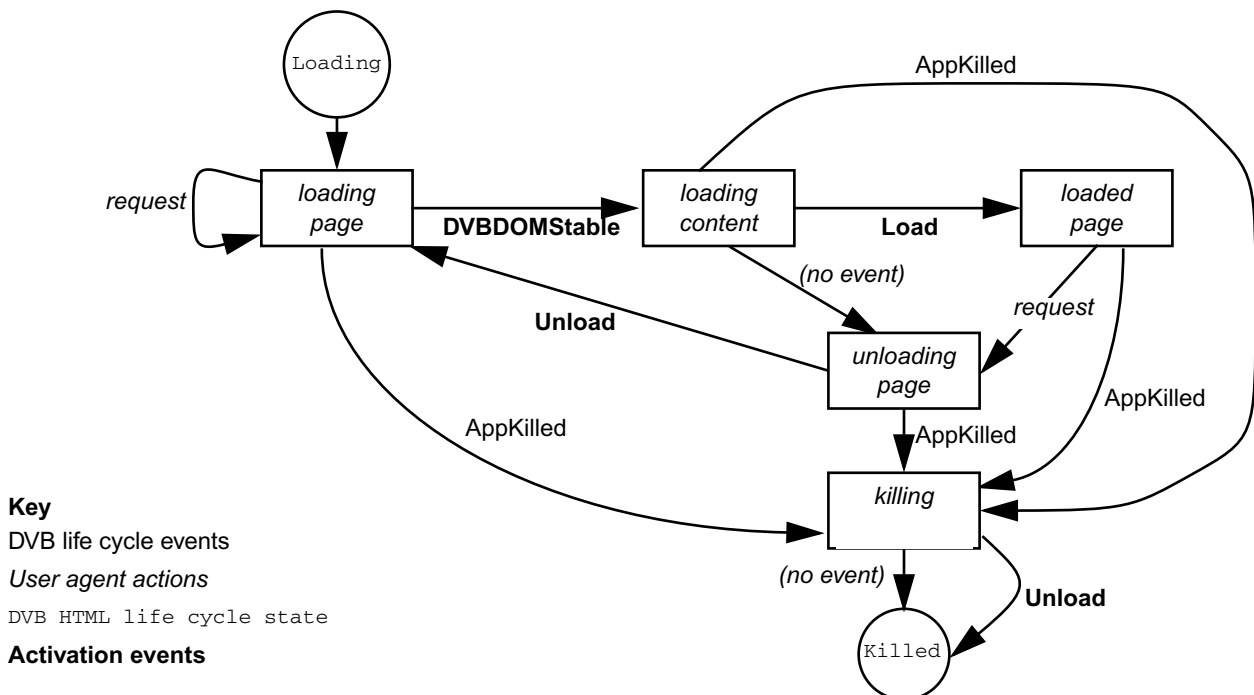


Figure 18 : Activation event state model

### 9.4.1 Event queue handling

The HTML and DOM standards say little about how events are inserted by the User Agent or how they are queued. DVB-HTML User Agents shall observe the following list of clarifications.

- After the `unload` event is sent, no other events shall be sent to the document, so all trigger event listeners may be discarded. (For clarification: `AppDestroyed` and `AppKilled` happen before this.)
- When navigating to a new page in a given window or frame, the user agent may load some or all of the new page and its associated files, before sending an `unload` to the current document. It shall not begin to render the new page, or deliver DOM events to it, until event handling of the `unload` event for the old page is complete.
- In loading resources for a new page, the user agent may register for, receive and queue DSM-CC events; however it shall not deliver them to the DOM until after it has delivered the `DVBDOMstable` event.
- The ECMAScript "document" variable shall be bound to the relevant document in all event handlers.

NOTE Together the above imply that trigger events (as is theoretically the case with user input events) may be lost during page transitions or even during presentation of a page. Authors should be aware of this possibility and code accordingly.

## 9.5 Inter application resource management

This is implementation dependent except as detailed below.

- Where there is a resource conflict between two applications signalled as part of the same service and running in the same service context, this shall be resolved using the priority signalled in the `application_priority` field in the application descriptor for each application.

## 9.6 Life cycle of Xlets embedded in DVB-HTML

The state of an Xlet is influenced by the state of the page that contains it, in addition to being influenced by the state of the containing application. The Xlet's lifecycle is bound by the lifecycle of the HTML application and page, but the Xlet may, through its own actions, be in a less active state than the containing HTML page. For example, an Xlet may destroy itself even if the page in which it is contained is active.

### 9.6.1 Starting embedded Xlets

For each Xlet specified with an object tag, a new classloader shall be created to load its classes. The standard MHP semantics surrounding Xlet classloaders apply. For example, classloaders shall not be shared with other Xlets.

When an HTML page is being loaded, any Xlets embedded within that page shall be requested to enter the loaded state, as defined for the `java.xlet.Xlet` interface.

Sometime before the page becomes visible, the platform shall request all embedded Xlets in that page to enter the paused state.

During the process of making a page visible, the platform shall request all paused Xlets (i.e. those that did not fail or voluntarily terminate before reaching this state) to enter the active state. The exact point during this process at which this happens is implementation dependent, but shall not be before any load event for the page.

When the containing DVB-HTML application transitions from the paused state to the active state, all paused Xlets that are on the current page shall be requested to enter the active state.

### 9.6.2 Termination

When a page becomes invisible due to navigation away from that page or when the containing DVB-HTML application becomes paused, all embedded Xlets that are in the active state shall be requested to enter the paused state.

It is implementation dependent when an Xlet that was paused due to navigation is transitioned from the paused state to the destroyed state, but this shall be after any unload event for the page in which it was embedded.

When the HTML application enters the killed state, all embedded Xlets must be put into the destroyed state by the platform.

When the DVB-HTML application enters the HTML destroyed state, it has no effect on the state of any embedded Xlets. Those embedded Xlets may choose to change their state as resources are withdrawn during such a transition, however they are only forced to change state when the DVB-HTML application enters the killed state.

The standard MHP rules concerning the destruction of an Xlet apply. Notably, an Xlet that fails to respond to a lifecycle notification is eligible for destruction, even if the containing page is still visible. Similarly, an Xlet can request that it be destroyed. This will cause the Xlet to be destroyed, but it will have no effect on the lifecycle of the page in which the Xlet is contained nor shall it cause the page to be reflowed. Similarly, an Xlet that puts itself in the paused state will have no effect on the state of the enclosing HTML page or application.

Of course, an Xlet might initiate an action that causes the application containing it to be terminated. For example, if the Xlet does a service selection, the application might not be signalled in the new service. Here, the existing MHP semantics concerning application lifecycle apply.

### 9.6.3 General issues

Multiple Xlets per XHTML document are allowed to the same extent as multiple Xlets are allowed in the same DVB service.

Instances of embedded Xlets:

- shall not be listed in the AIT
- shall not be exposed in the application listing an launching API when launched through a DVB HTML application
- inherit the transport protocols of the containing HTML application

When an embedded Xlet is in its active state, it shall have DOM access. It is implementation dependent if it will have DOM access in other states.

## 9.7 Services and applications not related to conventional DVB services

Section 9.1 of this specification describes MHP applications whose lifecycle is controlled through broadcast services. This specification additionally supports applications whose lifecycle is not bound to that of broadcast services. These include:

- Applications signalled through the interaction channel (see 9.7.1, "[Applications loaded from the interaction channel](#)" on page 204).
- Applications running from storage in the MHP terminal (see 9.7.2, "[Stored services](#)" on page 204).

### 9.7.1 Applications loaded from the interaction channel

In this scenario the application signalling comes from the interaction channel rather than the broadcast channel. In overview:

- Data equivalent to the broadcast AIT is provided from the interaction channel in the "[AIT file](#)". The connection to the server that provides the data is analogous to a connection to broadcast service.
- A locator specifying the location of the "[AIT file](#)" is analogous to the locator specifying a broadcast service.
- Navigation to this locator is analogous to selection of a broadcast service. The service selection could either be from the MHP terminal's intrinsic navigator or via an application using the service selection API.
- As with navigation between broadcast services when an interaction channel locator is selected as the current service the MHP terminal retrieves and decodes the "[AIT file](#)". Decisions about which currently running applications are allowed to remain running and which new applications are launched are based on the information in the "[AIT file](#)" just as they would be when navigating between broadcast services.
- When a locator specifying an AIT file is successfully used in a service selection operation, the applications database shall be populated with the information from that AIT file.

### 9.7.2 Stored services

Stored services are an encapsulation of one or more applications. The lifecycle of a stored service shall obey the same rules as defined for a broadcast service in 9.1.1, "[Basic lifecycle control](#)" on page 183. So, for example, currently running applications will be stopped if they are not listed to be runnable in the stored service's AIT information.

Hence there is no requirement to execute a stored services simultaneously with a broadcast service or application.

Stored applications running as part of a stored service shall:

- Execute with the [Application identification](#) that was stored with them when they were stored.
- Be able to access a file system using MHP APIs where:
  - any file that is present shall have the same name, relative path as it did in the broadcast file system.
  - any API providing byte level access to the data of a file shall return the same data as was broadcast.
  - the set stored may be a superset of the files listed in the [Application description file](#).

NOTE: This does not preclude the implementation also storing an optimised representation of files that would be presented to certain APIs. For example, this allows class files to be precompiled for subsequent presentation a the class loader.

The results of an application attempting to access files or directories above the directory that originally contained the [Application description file](#) are implementation dependant but shall either succeed or fail using the failure modes specified for the mechanism used.



When a stored service is started, any previously running streamed media decoders shall continue to run. It is the responsibility of applications to stop them if so required. Any resources used by such streamed media decoders shall have the lowest possible priority in any resource conflicts.

A stored service has the same lifecycle semantics as other services types (such as broadcast). So, when a stored service is selected the stored AIT information is inspected to determine which (if any) applications should auto start and which (if any) currently running applications are allowed to continue running by virtue of being included in the stored service.

NOTE: Currently, no API has been defined that would allow the addition of externally authorised applications to a stored service.

A stored service is built up by identifying applications from a broadcast service. It is not a copy of a broadcast service.

When an application is added to a stored service, the following information shall be stored:

- The [storage\\_property](#) and [version](#) information from the [Application storage descriptor](#) (see 10.14.2 on page 238).
- Sufficient information to be able to reconstruct the `application_descriptors_loop` (e.g. so that it can be returned by the `org.dvb.application.AppAttributes.getProperty()` method).
- The application type.
- The [Application identification](#).
- At least all those files listed in the [Application description file](#) with [priority](#) of "critical".
- If the [Application description file](#) lists any MHP security file (such as a signature file or a CRL) then these files shall be stored in the normal way (with due regard to the [priority](#) etc.). However, it is not necessary to list the security files in order to retain the security information as the method of retention of this information is an implementation detail.
- Implementations may store files in addition to those listed in the [Application description file](#).
- Sufficient information about the certificate chain that authenticated the application to be able to determine if any of those certificates have been revoked when it comes to run the application.
- Sufficient information to be able to construct the set of permissions to be granted to the application (this behaviour is equivalent to evaluating the permission request file at the time of launching the application).

Implications of the above are:

- If any non-critical files are needed for the application functionality then the application is responsible for obtaining them in the event that they are not stored.
- No information from AIT common loop descriptors is required to be stored.
- Information from the transport protocol descriptors is only relevant during the storing of the application. When the application is executing from storage it is responsible for creating the transport connections it requires.
- It is not required to store file MIME type as carried by the optional [Content type descriptor](#). The application developer is responsible for ensuring that stored files can be correctly interpreted without relying on this information.
- It is not required to store cache priority information from the [Caching priority descriptor](#). The application developer is responsible for ensuring that dynamic data is no listed in the [Application description file](#).

If applications within a stored service are also listed in a currently selected broadcast service they are runnable with the same constraints as other broadcast applications in the context of that broadcast service. So, as with "[Broadcast service related stored applications](#)", the consequence of storage is to allow better performance.

### 9.7.3 DVB-J Model

For DVB-J applications, APIs required to include some support for these applications include the following.

- "Service Selection API" (see 11.6.2 on page 259)
- "Content Referencing" (see 11.7.6 on page 266) as described in 11.11.12, "Support for the HTTP protocol in DVB-J" on page 280
- "Protocol Independent SI API" (see 11.6.5 on page 260) as described in 11.11.12, "Support for the HTTP protocol in DVB-J" on page 280

When a stored service is successfully created using the method `StoredServiceCreator.createStoredApplicationService`, this service shall appear in the list of services maintained by the `SIManager`. It shall be returned by `filterServices` both when passed an instance of `ServiceTypeFilter` constructed with the type `StoredApplicationService.STORED_APPLICATION_SERVICE` and when passed null to list all known services. The method `ServiceContext.select` shall support selecting stored services as identified by instances of `StoredApplicationService` created or returned using all these mechanisms. This is described in more detail in 11.12.2.2, "Modified behaviour of MHP 1.0 APIs" on page 281.

In profiles where application signalling over the interaction channel is supported, the method `SIManager.getService` shall accept instances of `javax.tv.locator.Locator` whose external form is valid "http" and "https" URLs and return a `Service` object. The method `ServiceContext.select` shall support selecting services identified by such service objects. This is described in more detail in 11.11.12, "Support for the HTTP protocol in DVB-J" on page 280.

### 9.7.4 Common behaviour

The environment in which stand-alone MHP applications are presented or executed shall be the same as for applications related to a conventional DVB service except as detailed below.

- Any existing video & audio running from before the application was started shall continue running. Applications shall not have a mechanism for attaining the identity of any such existing video or audio. Applications wishing to hide any such existing video may cover the full screen area with graphics. Applications have no ability to hide any such existing audio. MHP terminals shall assign the lowest possible resource priority to the presentation of any such existing video and audio.

## 9.8 Lifecycle of internet access applications

### 9.8.1 General issues

MHP terminals supporting the internet access profile shall support at least the execution of one internet access application at one time. There is no requirement to support the execution of more than one internet access application at one time. There is no requirement to support the execution of internet access applications and MHP applications at the same time. MHP terminals where internet access applications and MHP applications can be in memory and executing code at the same time but cannot share access to the screen are allowed. Such MHP terminals shall be considered as supporting internet access applications and MHP applications at the same time in the rest of this document. In these MHP terminals, any MHP applications in the active state may be put in the paused state when they loose access to the screen to an internet access application. This specification is intentionally silent about priorities for access to resources like memory in the event of a conflict between MHP applications and internet access applications.

### 9.8.2 Starting internet access applications from MHP applications

Where MHP terminals support simultaneous execution of internet access applications and MHP applications at the same time, then the internet access application shall start when requested. The internet access application shall be considered to be a peer of the MHP application and not a child of it. The lifecycle of the internet access application is not constrained by the lifecycle of the MHP application which started it.

Where MHP terminals do not support this simultaneous execution then the MHP application shall be destroyed / killed as part of the starting of the internet access application. In this case, when the end user of the MHP terminal exits the internet access application, the MHP terminal shall return to presenting the DVB service (if any) which it was presenting before the internet access application was started. Any MHP applications in this service will be re-started without any previous state using the normal mechanisms defined in version 1.0 of the MHP specification.

The APIs to enable DVB-J applications to manipulate internet access applications are defined in Annex AH, "(normative): Internet client APIs" on page 1365.

### 9.8.3 Starting MHP applications from internet access applications

In MHP, internet access applications shall support end user starting of DVB services (i.e. using the 'dvb:' URL) and MHP applications signalled over the interaction channel (i.e. using an AIT file referenced by an HTTP URL). This shall be handled by the normal service selection mechanism used elsewhere in the MHP specification (e.g. in the implementation of the service selection API).

On MHP terminals which support running internet access applications and MHP services at the same time, the new MHP application shall just start running. If there is already a running MHP service, that service shall be stopped as if the end user of the MHP terminal had selected the new service using the navigator. On MHP platforms which support running multiple MHP services at the same time, if no more MHP services can be started then it is implementation dependent which running service shall be stopped in order for the newly selected service to start.

On MHP terminals which do not support running internet access applications and MHP applications at the same time, the internet access application shall behave as if the end user of the MHP terminal had asked to exit that application. This may result in a platform specific dialogue with the end user, e.g. to ask about saving partly composed email messages as a draft. Depending on the nature of any such dialogue, the selection of the DVB service may fail if the end user of the MHP terminal does not wish to exit the internet access application concerned.

There is no requirement to remember the internet access application which selected a DVB service or to return to that internet access application when leaving the DVB service which it selected. This is the opposite of starting internet access applications from MHP applications.

## 9.9 Plug-ins

Section 5.4 of this specification defines the concept of plug-ins for use in migration situations, for the support of applications in existing content formats. Services and networks using such an approach to migration have a number of choices defined by this specification. Where a service carries one or more applications in an existing content format (referred to as delegated applications), those applications may be signalled either using the native signalling defined for that existing content format or using the AIT based signalling defined in this specification or both. Two types of plug-ins are defined: inter-operable plug-ins and implementation specific-plug-ins. This specification is intentionally silent about the operation and signalling of implementation specific plug-ins. Signalling of inter-operable plug-ins shall be done as defined in 10.13.2, "MHP signalling scenario" on page 236.

The lifecycle of inter-operable plug-ins shall obey the semantics of DVB-J applications concerning service selection operations as defined under the heading "A new service being selected replacing a previously selected one" in section 9.1.4.1 of this specification.

Where the AIT is used to signal delegated applications, the maintainer of the specification for that format shall register with the DVB project to obtain a value for the "application type" field of the AIT (see table 76, "Application types" on page 212). Having done this, the maintainer of the specification for the existing content format is responsible for defining any additional descriptors for the AIT which are required for the support of their content format.

---

# 10 Application Signalling

## 10.1 Introduction

This section covers the following topics:

- how the receiver identifies the applications associated with a service and finds the locations from which to retrieve them
- the signalling that enables the broadcast to manage the lifecycles of applications
- how the receiver can identify the sources of broadcast data required by the applications of a service

Much of the signalling is generic. For example, the [Application descriptor](#) is independent of the application representation. Other signalling is specific to the application representation or transport protocol (such as the [DVB-J application descriptor](#) and the [IP Routing Descriptors](#)).

### 10.1.1 Summary of common signalling

The minimum signalling requirements for any MHP applications are summarised as follows:

- PMT with [Application Signalling Descriptor](#) to identify the service component carrying the [Application Information Table](#).
- [Application Information Table](#) with the following information in its common descriptor loop:
  - [Transport protocol descriptor](#) (all applications descriptions shall be within the scope of at least one [Transport protocol descriptor](#). These can be placed in either or both of the descriptor loops)
- [Application Information Table](#) with the following information in its application information descriptor loop:
  - [Application descriptor](#)
  - [Application name descriptor](#)

### 10.1.2 Summary of additional signalling for DVB-J applications

The minimum additional signalling required for DVB-J applications are summarised as follows:

- [Application Information Table](#) with the following information in its application information descriptor loop:
  - [DVB-J application descriptor](#)
  - [DVB-J application location descriptor](#)

### 10.1.3 Summary of additional signalling for DVB-HTML applications

The minimum additional signalling required for DVB-HTML applications are summarised as follows:

- [Application Information Table](#) with the following information in its application information descriptor loop:
  - [DVB-HTML application descriptor](#)
  - [DVB-HTML application location descriptor](#)

### 10.1.4 Summary of additional signalling for applications carried via OC

In either the "common" (first) descriptor loop or the "application" (inner) descriptors loop:

- [Transport protocol descriptor](#), with the selector bytes containing the OC specific information as defined in table 94

### 10.1.5 Summary of additional signalling for applications carried via IP

[Application Information Table](#) with the following information in its common descriptor loop:

- [Routing Descriptor IPv4](#) or [Routing Descriptor IPv6](#) as appropriate.

In either the "common" (first) descriptor loop or the "application" (inner) descriptors loop:

- [Transport protocol descriptor](#), with the selector bytes containing the IP specific information as defined in [table 95](#).

### 10.1.6 How to add a new scheme (informative)

The signalling scheme is intended to be extensible with regard to the application representations and transport protocols that are supported. The areas that need to be addressed when doing this are summarised below.

To add further transport protocols:

- Extend [table 93](#), "[Semantic of selector bytes](#)" on [page 224](#)
- Possibly define further specialist descriptors such as the [IP Routing Descriptors](#)

To add further application representations:

- Define further specialist descriptors such as the [10.9](#), "[DVB-J specific descriptors](#)" on [page 230](#)
- Define the application type specific life cycle control codes in [10.6](#), "[Control of application life cycle](#)" on [page 215](#).

Where constant values are registered by this specification extend the [table 107](#), "[Registry of constant values](#)" on [page 234](#).

### 10.1.7 Service information

See [10.12](#), "[Service Information](#)" on [page 235](#).

## 10.2 Program Specific Information

The elementary stream (inner) loop of the PMT for a DVB service supporting one or more MHP applications must reference streams for the following:

- location of the stream transporting the [Application Information Table](#)
- location of the stream(s) transporting the application code and data

### 10.2.1 Application signalling stream

The elementary stream information for the PMT entry describing the elementary stream carrying the [Application Information Table](#) has the following characteristics:

- The `stream_type` is set to 0x05 (ITU-T Rec. H.222.0 | ISO/IEC 13818-1 private sections).
- An [Application Signalling Descriptor](#)

There may be more than one elementary stream carrying application signalling information for a service.

### 10.2.2 Data broadcast streams

The minimum signalling in the PMT associated with data broadcast components is the value of the PMT `stream_type` field required by the DVB data broadcasting specification ([EN 301 192 \[5\]](#)) for the transport protocol. The full details of the data broadcast protocol, the location of its "principal" component etc. are provided in the AIT (see [10.4](#), "[Application Information Table](#)" on [page 210](#)).

Optionally the PMT may include [Data broadcast id descriptors](#).

NOTE: Inclusion of **Data broadcast id descriptors** enables receivers to start mounting the file system that delivers applications concurrently with acquiring the AIT that identifies which applications are of interest. Enabling this concurrent operation may allow receivers to accelerate their activation of an interactive application. See B.2.8, "Mounting an Object Carousel" on page 411.

The **Data broadcast id descriptor** identifies the "principal" component of the data broadcast. The detailed semantics of this optional signalling reflects the transport protocol. For example, in the case of a DVB Object Carousel it identifies the component carrying the DSI.

There may also be certain protocol specific descriptors in the PMT. For example, the Object Carousel requires the inclusion of the **carousel\_id\_descriptor** (see B.2.8, "Mounting an Object Carousel" on page 411).

In its minimum form (with no selector information) a **Data broadcast id descriptor** just identifies the "principal" component. This optionally may be extended with selector information that identifies the application types of the autostart applications delivered by that data broadcast. See 10.7.2, "Data broadcast id descriptor" on page 217.

## 10.3 Notation

### 10.3.1 reserved

The term "reserved" when used in the clause defining the coded bit stream, indicates that the value may be used in the future for ISO defined extensions. Unless otherwise specified within the present chapter all "reserved" bits shall be set to "1".

### 10.3.2 reserved\_future\_use

The term "reserved\_future\_use", when used in the clause defining the coded bit stream, indicates that the value may be used in the future for ETSI defined extensions. Unless otherwise specified within the present chapter all "reserved\_future\_use" bits shall be set to "1".

## 10.4 Application Information Table

The **Application Information Table** (AIT) provides full information on the data broadcast, the required activation state of applications carried by it etc.

Data in the AIT allows the broadcaster to request that the receiver change the activation state of an application.

### 10.4.1 Data errors

AITs which contain errors shall be processed as follows:-

- An error in an application loop shall result in that entry in the application loop being silently discarded. Processing of that application loop shall continue with the next entry (if any).
- An error in an application information section outside of an application loop shall result in that entire application information section being silently discarded. Processing of the AIT shall continue with the next application information section (if any).

### 10.4.2 AIT transmission and monitoring

The minimum repetition rate for each AIT subtable is 10 seconds.

Provided that AITs for the selected service are delivered on 3 or fewer elementary streams then the maximum time interval between the moment the AIT is updated and the moment the new version is detected by the MHP shall be no more than 30 seconds.

Note If broadcasts use more than 3 elementary streams to deliver AITs then receiver response time may degrade in an unpredictable way.

The MHP terminal is only required to monitor AIT sections for application types that it can decode. The terminal's ability to decode also comprehends the interoperable plug-ins available to the terminal.

The set of application types listed in the application database reflects the set of AIT sections being monitored. So, this may be a subset of the application types being broadcast in the case that the broadcast carries a superset of the terminal's capabilities.

### 10.4.3 Optimised AIT signalling

The optional `AIT_version_number` carried by the `Application Signalling Descriptor` allows a possible optimisation of receiver burden as it allows receivers to acquire the AIT only after they see changes in the AIT version advertised in the PMT.

See 10.7.1, "Application Signalling Descriptor" on page 217.

### 10.4.4 Visibility of AIT

If an application tunes away from a transport stream where its signalling is carried without selecting a new service, it will continue running although the AIT is not visible.

In MHP terminals with multiple network interfaces, if the AIT of the selected service is visible via any of them, then the AIT signalling is used as normal.

### 10.4.5 Definition of sub-table for the AIT

All sections on the same PID with the AIT `table_id` and the same value of `application_type` are members of the same sub-table.

### 10.4.6 Syntax of the AIT

The Application Information Section describes applications and their associated information. Each Application Information Section includes one "common" descriptor loop at the top level for descriptors that are shared between applications of that sub table and a loop of applications. Each application in the application loop has an "application" descriptor loop containing the descriptors associated with that application.

Like DVB SI tables, the scope of common loop descriptors is the sub-table. So, any descriptors present in the common descriptor loop apply to all sections of the sub-table. Typically, common descriptors would normally only be present in section 0 of a sub-table, unless there was not enough space.

Like other DVB SI tables, any strings contained in these tables shall not have null terminations.

**Table 75 : Application Information Section syntax (Sheet 1 of 2)**

	No.of Bits	Identifier
<code>application_information_section() {</code>		
<code>table_id</code>	8	uimsbf
<code>section_syntax_indicator</code>	1	bslbf
<code>reserved_future_use</code>	1	bslbf
<code>reserved</code>	2	bslbf
<code>section_length</code>	12	uimsbf
<code>application_type</code>	16	uimsbf
<code>reserved</code>	2	bslbf
<code>version_number</code>	5	uimsbf
<code>current_next_indicator</code>	1	bslbf
<code>section_number</code>	8	uimsbf
<code>last_section_number</code>	8	uimsbf
<code>reserved_future_use</code>	4	bslbf
<code>common_descriptors_length</code>	12	uimsbf
<code>for (i=0; i&lt;N; i++) {</code>		
<code>descriptor()</code>		
<code>}</code>		
<code>reserved_future_use</code>	4	bslbf
<code>application_loop_length</code>	12	uimsbf



**Table 75 : Application Information Section syntax (Sheet 2 of 2)**

	No.of Bits	Identifier
<pre> for(i=0;i&lt;N;i++){   application_identifier()   application_control_code   reserved_future_use   application_descriptors_loop_length   for(j=0;j&lt;N;j++){     descriptor()   } } CRC_32 </pre>	8	uimsbf
	4	bslbf
	12	uimsbf
	32	rpchof

**table\_id:** This 8 bit integer with value 0x74 identifies this table.

**section\_syntax\_indicator:** The section\_syntax\_indicator is a 1-bit field which shall be set to "1".

**section\_length:** This is a 12-bit field, the first two bits of which shall be '00'. The remaining 10 bits specify the number of bytes of the section starting immediately following the section\_length field, and including the CRC\_32. The value in this field shall not exceed 1021 (0x3FD).

**application\_type:** This is a 16-bit field which identifies the type of the applications described in this AIT sub\_table. See table 76.

**Table 76 : Application types**

application_type	description
0x0000	reserved
0x0001	DVB-J application
0x0002	DVB-HTML application
0x0003...0xFFFF	subject to registration with DVB

**version\_number:** This 5-bit field is the version number of the sub\_table. The version\_number shall be incremented by 1 when a change in the information carried within the sub\_table occurs. When it reaches value "31", it wraps around to "0". When the current\_next\_indicator is set to "1", then the version\_number shall be that of the currently applicable sub\_table. When the current\_next\_indicator is set to "0", then the version\_number shall be that of the next applicable sub\_table.

**current\_next\_indicator:** This 1-bit indicator, when set to "1" indicates that the sub\_table is the currently applicable sub\_table. When the bit is set to "0", it indicates that the sub\_table sent is not yet applicable and shall be the next sub\_table to be valid.

**section\_number:** This 8-bit field gives the number of the section. The section\_number of the first section in the sub\_table shall be "0x00". The section\_number shall be incremented by 1 with each additional section with the same table\_id, and application\_type.

**last\_section\_number:** This 8-bit field specifies the number of the last section (that is, the section with the highest section\_number) of the sub\_table of which this section is part.

**common\_descriptors\_length:** This 12-bit field gives the total length in bytes of the following descriptors. The descriptors in this descriptor loop apply for all of the applications contained in this AIT sub\_table.

**application\_control\_code:** This 8-bit field controls the state of the application. The semantics of this field is application type dependant. See 10.6, "Control of application life cycle" on page 215.



**application\_loop\_length:** This 12-bit field gives the total length in bytes of the following loop containing application information.

**application\_identifier():** This 48 bit field identifies the application. The structure of this field is defined in 10.5, "Application identification" on page 214.

The same `application_identifier()` shall appear only once within an AIT subtable.

**application\_descriptors\_loop\_length:** This 12-bit field gives the total length in bytes of the following descriptors. The descriptors in this loop apply to the specific application.

**CRC\_32:** This is a 32-bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in annex B of EN 300 468 after processing the entire section.

### 10.4.7 Use of private descriptors in the AIT

Private descriptors may be included in the AIT provided that they are in the scope of a DVB-SI [4] private data specifier descriptor. The scope rules for the private data specifier descriptor are as follows:

- If this descriptor is located within any descriptor loop of the AIT, then any specifier identified within this descriptor loop applies to all following descriptors and user-defined values in the particular descriptor loop until the end of the descriptor loop, or until another occurrence of a `private_data_specifier_descriptor`.
- The use of the descriptor in the common (first) descriptor loop does not apply to descriptors or user-defined values in the application (second) descriptor loop.

### 10.4.8 Text encoding in AIT

Unless otherwise specified, all fields interpreted as text strings in the AIT shall be encoded as UTF8 (see 7.1.5, "Monomedia format for text" on page 66). See also 14.5, "Text encoding of application identifiers" on page 357.

### 10.4.9 AIT file

#### 10.4.9.1 Syntax

The interaction channel encoding of the AIT into the AIT file is as follows:

- A single file shall contain all of the data
- The file shall contain a concatenation of Application Information Sections (specified in MHP 1.0)
- The possibly multiple sections shall be ordered as follows:
  - ascending order of `application_type`
  - within a single value of `application_type` in ascending order of `section_number`
- All sections shall have `current_next_indicator` set to '1'

#### 10.4.9.2 Syntactic restrictions

##### 10.4.9.2.1 Transport protocols

The only allowed transport protocol type has id value 0x0003. See table 92, "Protocol\_id" on page 224.

##### 10.4.9.3 Semantics

The AIT file shall be loaded once during service selection. There is no requirement to monitor or poll the AIT file for any subsequent changes except as part of a subsequent service selection operation referring to the same AIT file. Consequences of this are:

- Only the AUTOSTART and PRESENT application control codes (of those defined in MHP 1.0) are appropriate.
- No standard mechanism for dynamic lifecycle control is provided.

NOTE: If dynamic lifecycle control is required application private mechanisms could be used. For example, a TCP connection over the interaction channel to a controlling server.

- Changes made to the AIT file after service selection shall not be detected or reported.

#### 10.4.9.4 MIME type

The MIME type for an AIT file shall be "application/dvb.ait". The file extension shall be ".ait".

## 10.5 Application identification

### 10.5.1 Encoding

Each application is associated with an application identifier. This is a 6 byte string with the following structure:

**Table 77 : Application identifier syntax**

	No. of Bits	Identifier	Value
application_identifier {			
organisation_id	32	bslbf	
application_id	16	bslbf	
}			

**organisation\_id:** This 32 bit field is a globally unique value identifying the organisation that is responsible for the application. These values are registered in [ETR 162 \[10\]](#).

This field is reproduced in the organisationName field of the subject name in the "leaf" certificate of an authenticated application (see [12.5.6, "subject" on page 295](#)).

Note The inclusion of this field in the leaf certificate provides authentication of the value.

**application\_id:** This 16 bit field uniquely identifies the application function. This is allocated by the organisation registered with the [organisation\\_id](#) who decides the policy for allocation within the organisation.

The application id values are divided into two ranges: one for unsigned applications and one for signed applications. This is for security reasons (see [12.1.1, "Overview of the security framework for applications" on page 286](#)). Applications transmitted as unsigned shall use an application id from the unsigned applications range and applications transmitted as signed shall use an application id from the signed applications range.

**Table 78 : Value ranges for application\_id**

application_id values	Use
0x0000...0x3fff	Application_ids for unsigned applications
0x4000...0x7fff	Application_ids for signed applications
0x8000...0xffffd	Reserved for future use by DVB
0xffffe	Special wildcard value for signed applications of an organisation
0xffff	Special wildcard value for all application of an organisation

Application id values 0xffff and 0xffffe are wild cards. They shall not be used to identify an application but, for example, are allowed for use in the [External application authorisation descriptor](#) see [10.7.5 on page 223](#). The value 0xffff matches all applications with the same [organisation\\_id](#). The value 0xffffe matches all signed applications with the same [organisation\\_id](#).

The same application identifier may be used in different application types for applications performing essentially the same function.

## 10.5.2 Effects on life cycle

The main concepts here are:

- On service change, currently running, previously broadcast, applications whose `service_bound_flag` is set to "0" shall (subject to resource restrictions) continue running if their application identifier is listed in the [Application Information Table](#) of the newly selected service.
- On service change, currently running, previously broadcast, applications whose `service_bound_flag` is set to "0" shall (subject to resource restrictions) continue running if their application identifier is suitably listed in the [External application authorisation descriptor](#) even if they are not part of the current service.
- Only a single instance of an application with a particular application identifier is allowed to execute at any time. So, if an application is already running then another instance of the same application shall not be launched. This affects the behaviour with respect to the application launching API and autostart applications after service selection.
- If the application signalling for an application has the "`service_bound_flag`" is set to "1", then the application is killed upon service selection.

See also S, "(normative): [Application Listing and Launching](#)" on page 711.

## 10.5.3 Authentication of application identification

See 12.5.6, "subject" on page 295.

# 10.6 Control of application life cycle

The broadcast signalling provides a mechanism for broadcasters to control the life cycle of standard application types. See also 9.1, "[Broadcast MHP applications](#)" on page 183.

## 10.6.1 Entering and leaving the domain of an application

The domain of an application is defined as the set of services where the application is listed in the AIT. This can be either as applications listed in the application (inner) loop of the AIT or as applications listed in the [External application authorisation descriptor](#). Services where the application is not listed in either of these two ways are outside of the domain of the application.

## 10.6.2 Dynamic control of the application life cycle

The dynamic control of the application life cycle is signalled through the [application\\_control\\_code](#) for the application in the AIT.

This control code allows the broadcaster to signal to the receiver what to do with the application with regard to its lifecycle. The set of codes have some differences between application types and so are defined on an application type specific basis.

If the receiver receives a code that it does not recognise the application shall continue in its current state.

When a change in these control codes causes a state change of a running MHP application, an `AppStateChangedEvent` shall be generated to all DVB-J applications which have registered to receive such events for the application concerned.

### 10.6.2.1 DVB-J

The application control codes for DVB-J applications are listed in 79.

**Table 79 : DVB-J application control code values**

code	identifier	semantics
0x00		reserved
0x01	AUTOSTART	The file system element(s) (e.g. an Object Carousel module) containing the class implementing the Xlet interface is loaded, The class implementing the Xlet is loaded into the VM and an Xlet object is instantiated, and the application is started subject to usual restrictions, etc.
0x02	PRESENT	Indicates that the application is present in the service, but is not autostarted.
0x03	DESTROY	When the control code changes from AUTOSTART or PRESENT to DESTROY, the destroy method of the Xlet is called (with the <b>unconditional</b> parameter set to <b>false</b> ) by the application manager and the application is allowed to destroy itself gracefully.
0x04	KILL	When the control code changes from AUTOSTART or PRESENT to KILL, the destroy method of the Xlet is called (with the <b>unconditional</b> parameter set to <b>true</b> ) by the application manager.
0x05	reserved	
0x06	REMOTE	This identifies a remote application that is only launchable after tuning.
0x07...0xFF		reserved for future use

See 9.2.3, "DVB-J Application Lifecycle" on page 188.

### 10.6.2.2 DVB-HTML

The application control codes for DVB-HTML applications are listed in 80.

**Table 80 : DVB-HTML application control code values (Sheet 1 of 2)**

code	identifier	semantics
0x00		reserved
0x01	AUTOSTART	The <a href="#">Application Entry Point</a> of the DVB-HTML application is loaded. This is loaded into the user agent, and the DVB-HTML actor is created (in the <a href="#">Loading</a> state) and the DVB-HTML application is started. When these steps are complete the DVB-HTML actor is in the <a href="#">Active</a> state.
0x02	PRESENT	Indicates that the DVB-HTML application is present in the service, but is not autostarted.
0x03	DESTROY	When the control code changes from AUTOSTART or PRESENT to DESTROY, the DVB-HTML actor goes to the <a href="#">Killed</a> state.
0x04	KILL	When the control code changes from AUTOSTART or PRESENT to KILL the DVB-HTML actor is terminated.
0x05	PREFETCH	As for AUTOSTART except that the DVB-HTML actor holds on entry to the <a href="#">Active</a> state and waits for a trigger before completely transitioning to the <a href="#">Active</a> state.

**Table 80 : DVB-HTML application control code values (Sheet 2 of 2)**

code	identifier	semantics
0x06	REMOTE	This identifies a remote application that is only launchable after tuning.
0x07...0xFF		reserved for future use

See 9.3.2, "DVB-HTML Application Lifecycle" on page 195.

## 10.7 Generic descriptors

### 10.7.1 Application Signalling Descriptor

The application signalling descriptor is defined for use in the elementary stream loop of the PMT where the `stream_type` of the elementary stream is 0x05. It identifies that the elementary stream carries an [Application Information Table](#).

The application signalling descriptor optionally carries a loop of `application_type` and `version_number` pairs. These allow the descriptor to optionally reproduce the current version number state of the associated [Application Information Table](#). This allows the receiver to be informed of the version of the AIT as a side effect of monitoring the PMT (which is expected to be monitored closely, under normal conditions). See 10.4.3, "Optimised AIT signalling" on page 211.

When the MHP detects a change of the content of the application signalling descriptor, it shall acquire the new version of the AIT and respond accordingly.

The presence of the `application_type` and `AIT_version` subfields is optional. If not present then the [AIT transmission and monitoring](#) applies, see 10.4.2, "AIT transmission and monitoring" on page 210.

**Table 81 : application signalling descriptor syntax**

	No.of Bits	Identifier
<code>application_signalling_descriptor() {</code>		
<code>descriptor_tag</code>	8	uimsbf
<code>descriptor_length</code>	8	uimsbf
<code>for( i=0; i&lt;N; i++ ){</code>		
<code>application_type</code>	16	uimsbf
<code>reserved_future_use</code>	3	bslbf
<code>AIT_version_number</code>	5	uimsbf
<code>}</code>		
<code>}</code>		

**descriptor\_tag:** This 8 bit integer with value 0x6F identifies this descriptor.

**descriptor\_length:** This 8 bit field indicates the number of bytes following the descriptor length field.

**application\_type:** This 16 bit field identifies the application type of an [Application Information Table](#) sub-table that is on this elementary stream.

**AIT\_version\_number:** This 5 bit field provides the "current" version number of the [Application Information Table](#) sub-table identified by the application type field.

### 10.7.2 Data broadcast id descriptor

The data broadcast id descriptor is defined for use in the elementary stream information of the PMT. The descriptor identifies:

- the transport format of the data broadcast whose "principal component" is on this elementary stream.  
The semantics of "principal component" is transport protocol specific.
- the set of application types for any autostart applications delivered by the data broadcast.

For a single elementary stream more than one data broadcast id descriptor may be used to list additional applications types, however, each descriptor shall indicate the same data broadcast id.

More than one elementary stream may have a data broadcast id descriptor indicating that auto start applications are carried by more than one delivery mechanism (for example a single service may have more than one object carousel delivering auto start applications).

### 10.7.2.1 Generic descriptor

The data broadcast id descriptor is defined in a generic form by the DVB SI-DAT specification (illustrated in table 82). Where no "id specific data" is provided the descriptor just identifies the "principal" component of a data broadcast.

**Table 82 : generic data broadcast id descriptor syntax**

	No.of Bits	Identifier	Value
<code>data_broadcast_id_descriptor() {</code>			
<code>descriptor_tag</code>	8	uimsbf	
<code>descriptor_length</code>	8	uimsbf	
<code>data_broadcast_id</code>	16	uimsbf	
for (i=0; i<N; i++) {			
<code>id specific data</code>	8	bslbf	
}			
}			

### 10.7.2.2 MHP data broadcast id descriptor

When the data broadcast id is one of those defined by this specification (see table 107) the syntax of the data broadcast id descriptor is as shown in table 83. This allows the data broadcast id descriptor to provide information about the autostart applications that exist within the data broadcast.

**Table 83 : MHP data broadcast id descriptor syntax**

	No.of Bits	Identifier	Value
<code>data_broadcast_id_descriptor() {</code>			
<code>descriptor_tag</code>	8	uimsbf	
<code>descriptor_length</code>	8	uimsbf	
<code>data_broadcast_id</code>	16	uimsbf	
for (i=0; i<N; i++) {			
<code>application_type</code>	16	uimsbf	
}			
}			

**descriptor\_tag:** This 8 bit integer with value 0x66 identifies this descriptor.

**data\_broadcast\_id:** This 16 bit field indicates the format of the data broadcast transport protocol. These values are registered in ETR 162.

**application\_type:** This 16 bit field indicates the type of the application (i.e. the engine or plug-in on which the application can be executed). See table 76 on page 212.

### 10.7.3 Application descriptor

Exactly one instance of the application descriptor shall be contained in every "application" (inner) descriptor loop of the AIT.

**Table 84 : application descriptor syntax**

	No.of Bits	Identifier	Value
application_descriptor() {			
descriptor_tag	8	uimsbf	
descriptor_length	8	uimsbf	
application_profiles_length	8	uimsbf	
for( i=0; i<N; i++ ) {			
application_profile	16	uimsbf	
version.major	8	uimsbf	
version.minor	8	uimsbf	
version.micro	8	uimsbf	
}			
service_bound_flag	1	bslbf	
visibility	2	bslbf	
reserved_future_use	5	bslbf	
application_priority	8	uimsbf	
for( i=0; i<N; i++ ) {			
transport_protocol_label	8	uimsbf	
}			
}			

**descriptor\_tag:** This 8 bit integer with value 0x00 identifies this descriptor.

**application\_profiles\_length:** This 8-bit field indicates the length of the application\_profile loop in bytes.

**application\_profile:** This 16 bit field is an integer value which represents the application type specific profile. This indicates that a receiver implementing one of the profiles listed in this loop is capable of executing the application.

**version.major:** This 8 bit field carries the numeric value of the major sub-field of the profile version number.

**version.minor:** This 8 bit field carries the numeric value of the minor sub-field of the profile version number.

**version.micro:** This 8 bit field carries the numeric value of the micro sub-field of the profile version number.

The four above fields indicate the minimum profile on which an application will run. Applications may test for features found in higher (backwards compatible) profiles and exploit them. The MHP terminal shall only launch applications if the following expression is true for at least one of the signalled profiles:

$$\begin{aligned}
 & (\text{application\_profile} \in \text{terminal\_profiles\_set}) \\
 & \wedge \{ (\text{application\_version.major} < \text{terminal\_version.major}(\text{application\_profile})) \\
 & \quad \vee [ (\text{application\_version.major} = \text{terminal\_version.major}(\text{application\_profile})) \\
 & \quad \wedge ( \{ \text{application\_version.minor} < \text{terminal\_version.minor}(\text{application\_profile}) \} \\
 & \quad \quad \vee \{ [\text{application\_version.minor} = \text{terminal\_version.minor}(\text{application\_profile})] \\
 & \quad \quad \quad \wedge [\text{application\_version.micro} \leq \text{terminal\_version.micro}(\text{application\_profile})] \} ) ] \}
 \end{aligned}$$

Where:

- ∈ represents "belongs to the set of"
- ∧ represents "logical AND"
- ∨ represents "logical OR"

See table 140, "Profile encoding" on page 365 for the encoding of these values.

**service\_bound\_flag:** If this field is set to "1", the application is only associated with the current service and so the process of killing the application shall start at the beginning of the service change regardless of the contents of the destination AIT.

**visibility:** This 2 bit field specifies the intended visibility of the application for possible presentation to the user for possible launching. Table 85 lists the allowed values of this field.

**Table 85 : Definition of visibility states for applications**

visibility	description
00	This application shall not be visible either to applications via an application listing API or to users via the navigator with the exception of any error reporting or logging facility, etc.
01	This application shall not be visible to users but shall be visible to applications via an application listing API.
10	Reserved
11	This application can be visible to users and shall be visible to applications via the application listing API.

**application\_priority:** This field identifies a relative priority between the applications signalled in this service.

- Where there is more than one application with the same [Application identification](#) this priority shall be used to determine which application is started.
- Where there are insufficient resources to continue running a set of applications, this priority shall be used to determine which applications to stop or pause.
- A larger integer value indicates higher priority.
- If two applications have the same application identification and the same priority, the MHP may make an implementation-dependent choice on which to start.

**transport\_protocol\_label:** This 8-bit field identifies a transport protocol that delivers the application. See [transport\\_protocol\\_label](#) in [Transport protocol descriptor](#).

If more than one protocol is signalled then each protocol is an alternative delivery mechanism. The ordering indicates the broadcaster's view of which transport connection will provide the best user experience (first is best). This may be used as a hint by MHP terminal implementations. It shall be evaluated only once during the life time of the application.

The protocol selection by the MHP terminal may depend on a variety of factors including user preferences and the performance of the transport connections to the terminal.

## 10.7.4 User information descriptors

The user information descriptors complement the "[Application descriptor](#)" by providing information suitable for presentation to the user (where the "[Application descriptor](#)" provides technical information for automatic use by the receiver).

These descriptors are defined for use in the application loop of the AIT.



### 10.7.4.1 Application name descriptor

Exactly one instance of this descriptor shall be included in the application information of an application. The application name shall distinguish the application and shall be informative to the user.

**Table 86 : application name descriptor syntax**

	No.of Bits	Identifier	Value
application_name_descriptor() {			
descriptor_tag	8	uimsbf	
descriptor_length	8	uimsbf	
for (i=0; i<N; i++) {			
ISO_639_language_code	24	bslbf	
application_name_length	8	uimsbf	
for (i=0; i<N; i++) {			
application_name_char	8	uimsbf	
}			
}			
}			

**descriptor\_tag:** This 8 bit integer with value **0x01** identifies this descriptor.

**ISO\_639\_language\_code:** This 24-bit field contains the **ISO 639.2 [19]** three character language code of the language of the following bouquet name. Both ISO 639.2/B and ISO 639.2/T may be used.

Each character is coded into 8 bits according to **ISO 8859 [20]** and inserted in order into the 24-bit field.

**application\_name\_length:** This 8 bit unsigned integer specifies the number of bytes in the application name.

**application\_name\_char:** This field carries a string (not null terminated) of characters encoded in accordance with annex A of ETS 300 468. The string names the application in a manner intended to be informative to the user.

### 10.7.4.2 Application icons descriptor

Zero or one instance of this descriptor shall be included in the application information of an application. It allows icons to be associated with the application. The content format for these possible icons shall be restricted PNG as specified in section 15.1, "PNG - restrictions" on page 363.

**Table 87 : application icons descriptor syntax**

	No.of Bits	Identifier	Value
application_icons_descriptor() {			
descriptor_tag	8	uimsbf	
descriptor_length	8	uimsbf	
icon_locator_length	8	uimsbf	
for (i=0; i<N; i++) {			
icon_locator_byte	8	uimsbf	
}			
icon_flags	16	bslbf	
for (i=0; i<N; i++) {			
reserved_future_use	8	bslbf	
}			
}			

**descriptor\_tag:** This 8 bit integer with value **0x0B** identifies this descriptor.

**icon\_locator\_length:** This 8 bit integer specifies the number of characters in the string that prefixes standard icon file name.

**icon\_locator\_byte:** This 8 bit value is one byte of the icon locator string.

The icon locator is the first part of the string that specifies the location of the icon files. This is application type dependant. See table 88. The `icon_locator` shall not end with a "/" slash character.

**Table 88 : Icon locator semantics**

application_type	description
0x0000	reserved
0x0001	For DVB-J this is a path relative to the base directory of the application as defined in 10.9.2, "DVB-J application location descriptor" on page 230.
0x0002	For DVB-HTML this is a path relative to the physical root of the application as defined in 10.10.2, "DVB-HTML application location descriptor" on page 232.
0x0003...0xFFFF	

**icon\_flags:** This 16 bit field carries a value which is the bitwise OR of the flag bits that identify the icons that are provided for the application. The flag bits are defined in table 89.

**Table 89 : Definition of different icon flags**

Icon flag bits	Description of icon size and pixel aspect ratio
0000 0000 0000 0001	32 x 32 for square pixel display
0000 0000 0000 0010	32 x 32 for broadcast pixels on 4:3 display (note 1)
0000 0000 0000 0100	24 x 32 for broadcast pixels on 16:9 display
0000 0000 0000 1000	64 x 64 for square pixel display
0000 0000 0001 0000	64 x 64 for broadcast pixels on 4:3 display
0000 0000 0010 0000	48 x 64 for broadcast pixels on 16:9 display
0000 0000 0100 0000	128 x 128 for square pixel display
0000 0000 1000 0000	128 x 128 for broadcast pixels on 4:3 display
0000 0001 0000 0000	96 x 128 for broadcast pixels on 16:9 display
xxxx xxx0 0000 0000	Reserved for future standardisation
NOTE 1: approx. 15/16 pixel aspect ratio on 50 Hz system	

The file names for the icon files are encoded in a standard way:

```
filename      = icon_locator "/"dvb.icon." hex_string
hex_string    = 4*4hex
hex           = digit | "A" | "B" | "C" | "D" | "E" | "F" | "a" | "b" | "c" | "d" | "e" | "f"
digit        = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

Where `hex_string` is the 4 digit hexadecimal representation of the value carried by `icon_flags`.

## 10.7.5 External application authorisation descriptor

The "common" (first) descriptor loop of the [Application Information Table](#) may contain zero or more external\_application\_authorisation\_descriptors. Each descriptor contains information about external applications that are allowed to continue to run with the applications listed in this [Application Information Table](#) sub-table but cannot be launched from this service. The external authorization applies to applications with the identified application\_identifier() that are of the application\_type identified by the AIT subtable where this descriptor is contained.

**Table 90 : external application authorisation descriptor syntax**

	No.of Bits	Identifier	Value
external_application_authorisation_descriptor() {			
descriptor_tag	8	uimsbf	
descriptor_length	8	uimsbf	
for(i=0; i<N; i++) {			
application_identifier()			
application_priority	8	uimsbf	
}			
}			

**descriptor\_tag:** This 8-bit integer with value `0x05` identifies this descriptor.

**application\_identifier():** This 48-bit field identifies an application. The structure of this field is defined in [10.5](#), "Application identification" on page 214.

**application\_priority:** This 8-bit integer specifies the priority that this application assumes in the context of the current service.

If the `0xffff` or `0xfffe` wildcard is used for the `application_id` within the `application_identifier()` and there are some applications from the same `organisation_id` explicitly signalled in the application loop of the AIT, the priority for those applications shall be the one signalled in the application\_descriptor (see [10.7.3](#) on page 219).

See `application_priority` under [10.7.3](#), "Application descriptor" on page 219.

## 10.8 Transport protocol descriptors

### 10.8.1 Transport protocol descriptor

The transport protocol descriptor identifies the transport protocol associated with a service component and possibly provides protocol dependent information.

The descriptor may be used in either the "common" (first) descriptor loop or the "application" (inner) descriptors loop. When in the "common" loop it applies to all of the applications in that sub-table. Any such descriptors in the "application" loop describe additional transport protocols available to a specific application.

Each application described in this section shall be in the scope of at least one transport protocol descriptor.

**Table 91 : transport protocol descriptor syntax**

	No.of Bits	Identifier	Value
transport_protocol_descriptor() {			
descriptor_tag	8	uimsbf	
descriptor_length	8	uimsbf	
protocol_id	16	uimsbf	
transport_protocol_label	8	uimsbf	
for(i=0; i<N; i++) {			
selector_byte	8	uimsbf	N1
}			
}			

**descriptor\_tag:** This 8 bit integer with value 0x02 identifies this descriptor.

**protocol\_id:** An identifier of the protocol used for carrying the applications. The values of the protocol\_id are be registered here and in ETR162.

**Table 92 : Protocol\_id**

protocol_id	description
0x0000	reserved
0x0001	MHP Object Carousel as defined in annex B, "(normative): Object carousel" on page 388.
0x0002	IP via DVB multiprotocol encapsulation as defined in EN 301 192, TR 101 202
0x0003	Transport via HTTP over the interaction channel as described in 10.8.1.3, "Transport via interaction channel" on page 226.
0x0004...0x00FF	Reserved for use by DVB
0x0100...0xFFFF	Subject to registration in ETR162

**transport\_protocol\_label:** Thus 8 bit field uniquely identifies a transport protocol within this AIT section. The Application descriptor refers to this value to identify a transport connection that carries the application.

**selector\_byte:** Additional protocol specific information.

**Table 93 : Semantic of selector bytes**

protocol_id	selector byte data
0x0000	reserved
0x0001	See 10.8.1.1 "Transport via OC".
0x0002	See 10.8.1.2 "Transport via IP".
0x0003	See 10.8.1.3 "Transport via interaction channel".
0x0004...0xFFFF	TBD

### 10.8.1.1 Transport via OC

When the protocol ID is 0x0001 the selector bytes in the Transport protocol descriptor shall be as shown in table 94.

**Table 94 : Syntax of selector bytes for OC transport**

Syntax	Bits	Mnemonic
<code>remote_connection</code>	1	bslbf
<code>reserved_future_use</code>	7	bslbf
<code>if( remote_connection == "1" ) {</code>		
<code>original_network_id</code>	16	uimsbf
<code>transport_stream_id</code>	16	uimsbf
<code>service_id</code>	16	uimsbf
<code>}</code>		
<code>component_tag</code>	8	uimsbf

**component\_tag:** Identifies the "principal" service component that delivers the application. The identified component is the elementary stream that carries the DSI of the object carousel.

**remote\_connection:** This single bit flag if set to "1" indicates that the transport connection is provided by a service that is different to the one carrying the AIT. When set the following 3 fields (`original_network_id`, `transport_stream_id` and `service_id`) are included in the selector bytes. This flag shall be set to "0" when the transport connection is provided by the current service.

Such applications shall not be autostarted by receivers but are visible (subject to the visibility field of the application descriptor) via an application listing API for possible launching by service selection (but not via an application launching API).

Applications with this flag set shall have their application control code set to REMOTE (see table 79 on page 216 and 80 on page 216).

See 11.7.2, "Application discovery and launching APIs" on page 261.

**original\_network\_id:** This 16 bit field identifies the DVB-SI original network id of the transport stream that provides the transport connection.

**transport\_stream\_id:** This 16 bit field identifies the MPEG transport stream id of the transport stream that provides the transport connection.

**service\_id:** This 16 bit field identifies the DVB-SI service id of the service that provides the transport connection.

### 10.8.1.2 Transport via IP

When the protocol ID is 0x0002 the selector bytes in the *Transport protocol descriptor* shall be as shown in table 95.

This structure includes two important components of the *data\_broadcast\_descriptor* defined in EN 301 192 [5]. It provides all the information necessary for the MHP to acquire applications and application data components delivered by IP protocols. The profiles where this is an optional or mandatory feature are listed in 15, "Detailed platform profile definitions" on page 361.

**Table 95 : Syntax of selector bytes for IP transport**

Syntax	Bits	Mnemonic
<code>remote_connection</code>	1	bslbf
<code>reserved_future_use</code>	7	bslbf
<code>if( remote_connection == "1" ) {</code>		
<code>original_network_id</code>	16	uimsbf
<code>transport_stream_id</code>	16	uimsbf
<code>service_id</code>	16	uimsbf
<code>}</code>		
<code>alignment_indicator</code>	1	bslbf
<code>reserved_future_use</code>	7	bslbf
<code>for( i=0; i&lt;N; i++){</code>		
<code>URL_length</code>	8	uimsbf
<code>for(j=0; j&lt;URL_length; j++){</code>		
<code>URL_byte</code>	8	uimsbf
<code>}</code>		
<code>}</code>		

**remote\_connection:** This and the associated 3 fields (`original_network_id`, `transport_stream_id` and `service_id`) have identical syntax and semantics to the fields with the same names under 10.8.1.1, "Transport via OC" on page 224.

**alignment\_indicator:** This 1-bit field indicates the alignment that exists between the bytes of the `datagram_section` and the Transport Stream bytes (equivalent to the field with this name defined in the EN 301 192 [5] MPE `data_broadcast_descriptor`).

**URL\_length:** This 8-bit field indicates the number of bytes in the URL.

**URL\_byte:** These bytes form a URL conforming to RFC 2396 [41].

For URL using the "server" field including the host:port notation as defined in RFC 2396 [41], only numeric IP addresses shall be used for identifying IP transmissions carried in the broadcast channel as there is no Domain Name Service in the broadcast-only scenario to be used for resolving names.

IP to MAC mapping shall be done as described in [RFC 1112 \[45\]](#).

### 10.8.1.3 Transport via interaction channel

When the protocol ID is 0x0003 the selector bytes in the [Transport protocol descriptor](#) shall be as shown in [table 95](#), "[Syntax of selector bytes for IP transport](#)" on [page 225](#). This allows encoding of a number of URLs. For efficiency when encoding possibly many similar URLs the encoding divides the URL into a shared base part and a set of URL extensions. The set of URLs can identify ZIP ([ZIP \[77\]](#)) files, or base URLs ending in the "/" character, that encapsulate portions of the file system. See [6.4.1.1](#), "[File system logical structure](#)" on [page 60](#) for a description of this file system.

Multiple transport protocol descriptors with the protocol ID value 0x0003 and the same transport protocol label may be provided to define a larger set of URLs to describe the file system.

**Table 96 : Syntax of selector bytes for interaction transport**

Syntax	Bits	Mnemonic
for( i=0; i<N; i++){ URL_base_length	8	uimsbf
for( j=0; j<N; j++){ URL_base_byte	8	uimsbf
}		
for( j=0; j<N; j++){ URL_length	8	uimsbf
for(k=0; k<URL_length; k++){ URL_byte	8	uimsbf
}		
}		
}		

**URL\_base\_length:** This 8-bit field provides the number of bytes in the base part of the URL.

**URL\_base\_byte:** These bytes form the first part of a HTTP URL conforming to HTTP 1.0 (see [RFC 1945 \[78\]](#)), or the first part of an HTTPS URL conforming to HTTPS (see [RFC 2818 \[98\]](#)). Where an HTTP URL is found, http shall be used as in [6.3.7.2](#), "[MHP profile of HTTP 1.0](#)" on [page 58](#). Where an HTTPS URL is found, https shall be used as in [6.3.7.3](#), "[HTTPS](#)" on [page 59](#).

**URL\_extension\_length:** This 8-bit field indicates the number of bytes in the extension part of the URL.

**URL\_extension\_byte:** These bytes form the later part of an HTTP URL conforming to HTTP 1.0 (see [RFC 1945 \[78\]](#)), or the later part of an HTTPS URL conforming to [HTTPS 6.3.7.3](#), "[HTTPS](#)" on [page 59](#).

URLs are formed by concatenating the URL extension with the preceding URL base. The URL so formed either identifies a file system directory or a specific ZIP file. See [6.4.1.1](#), "[File system logical structure](#)" on [page 60](#).

## 10.8.2 IP Routing Descriptors

The routing descriptors are defined for use in the "common" descriptor loop of the AIT. They indicate the service component location of packets for specified multicast IP addresses. The routing descriptors shall provide a complete list of all multicast IP addresses that applications in that AIT section may require.

Routing descriptors are mandatory where multicast IP is used in a service. Multiple descriptors may be used.

Two similar descriptors are defined, one for IPv4 and one for IPv6.

## 10.8.2.1 Routing Descriptor IPv4

Table 97 : Syntax of the IPv4 routing descriptor

Syntax	Bits	Mnemonic
<code>routing_descriptor_ip4 () {</code>		
<code>descriptor_tag</code>	8	uimsbf
<code>descriptor_length</code>	8	uimsbf
<code>for(i=0; i&lt;N; i++ ) {</code>		
<code>component_tag</code>	8	uimsbf
<code>address</code>	32	uimsbf
<code>port_number</code>	16	uimsbf
<code>address_mask</code>	32	uimsbf
<code>}</code>		
<code>}</code>		

**descriptor\_tag:** This 8-bit field with value `0x06` identifies this descriptor.

**descriptor\_length:** This 8-bit field identifies the number of bytes following the length field.

**component\_tag:** This 8-bit field identifies the service component that carries multicast IP packets for the specified IP address set and port.

An IP address is carried in this service component if the result of AND-ing that address with the `address_mask` field is equal to the `address` field.

**address:** This 32-bit field carries an IPv4 IP address.

**port\_number:** This 16-bit field carries a port number. The value zero is a "wild card" value and indicates any port number.

**address\_mask:** This 32-bit field carries a mask to be applied to the address field.

## 10.8.2.2 Routing Descriptor IPv6

Table 98 : Syntax of the IPv6 routing descriptor

Syntax	Bits	Mnemonic
<code>routing_descriptor_ip6 () {</code>		
<code>descriptor_tag</code>	8	uimsbf
<code>descriptor_length</code>	8	uimsbf
<code>for(i=0; i&lt;N; i++ ) {</code>		
<code>component_tag</code>	8	uimsbf
<code>address</code>	128	uimsbf
<code>port_number</code>	16	uimsbf
<code>address_mask</code>	128	uimsbf
<code>}</code>		
<code>}</code>		

**descriptor\_tag:** This 8-bit field with value `0x07` identifies this descriptor.

**descriptor\_length:** This 8-bit field identifies the number of bytes following the length field.

**component\_tag:** This 8-bit field identifies the service component that carries multicast IP packets for the specified IP address set and port.

An IP address is carried in this service component if the result of AND-ing that address with the `address_mask` field is equal to the `address` field.

**address:** This 128-bit field carries an IPv6 IP address.

**port\_number:** This 16-bit field carries a port number. The value zero is a "wild card" value and indicates any port number.

**address\_mask:** This 128-bit field carries a mask to be applied to the address field.

## 10.8.3 Pre-fetch signalling

### 10.8.3.1 Introduction

This signalling is defined to enable implementations to start fetching files that will be required during the early part of an application's life. Later in an applications' life it can actively request file pre-fetching using API mechanisms. Descriptors in this section do not have a relation to the API-based pre-fetching for this version of this specification.

This signalling is optional to broadcast and optional for implementations to consider.

### 10.8.3.2 Pre-fetch descriptor

Zero or one pre-fetch descriptors can be included in the "application" (inner) descriptor loop of the AIT. It is defined for use where the `protocol_id` of the transport is `0x0001` (MHP Object Carousel). Each descriptor is associated with a specific [Transport protocol descriptor](#) via the `transport_protocol_label`.

MHP terminals may use this descriptor to improve application start-up time by pre-fetching modules that have the indicated labels (see "[Label descriptor](#)" on page 390).

**Table 99 : Syntax of the pre-fetch descriptor**

Syntax	Bits	Mnemonic
<code>prefetch_descriptor () {</code>		
<code>descriptor_tag</code>	8	uimsbf
<code>descriptor_length</code>	8	uimsbf
<code>transport_protocol_label</code>	8	uimsbf
<code>for(i=0; i&lt;N; i++ ) {</code>		
<code>label_length</code>		
<code>for(j=0; j&lt;label_length; j++ ) {</code>		
<code>label_char</code>	8	uimsbf
<code>}</code>		
<code>prefetch_priority</code>	8	uimsbf
<code>}</code>		
<code>}</code>		

**descriptor\_tag:** This 8-bit field with value `0x0C` identifies this descriptor.

**descriptor\_length:** This 8-bit field identifies the number of bytes following the descriptor length field.

**transport\_protocol\_label:** This 8-bit field identifies the [Transport protocol descriptor](#) that specifies the transport connection that delivers the modules to which this prefetch descriptor refers. See "[transport\\_protocol\\_label](#)" on page 220.

**label\_length:** This 8-bit field identifies the number of bytes in the module label.

**label\_char:** These 8-bit fields carry an array of bytes that are a module label. This label matches a label on one or more module carried by [Label descriptors](#) in the `userInfo` fields of the `moduleInfo` structure of DIIs (see "[Label descriptor](#)" on page 390).

The same module label may be attached to several modules.

**prefetch\_priority:** A value between 1 and 100 (both inclusive). It expresses a pre-fetching hint of the modules with the corresponding label using the specified priority (100 highest, 1 lowest).



### 10.8.3.3 DII location descriptor

For each application zero or one DII location descriptors can be provided. It can be located in either the "common" (first) or "application" (inner) descriptor loop of the AIT. It is defined for use where the `protocol_id` of the transport is `0x0001` (MHP Object Carousel). Each descriptor is associated with a specific [Transport protocol descriptor](#) via the `transport_protocol_label`.

The modules that are part of a DSM-CC object carousel are signalled in DownloadInfoIndication (DII) messages. The object carousel does not list all the existing DII messages in a single place.

In order to find all of the modules that match a particular pre-fetch label (see [10.8.3.2, "Pre-fetch descriptor" on page 228](#)), it is necessary that all the relevant DII messages can be found. The DII location descriptor lists the locations of these DII.

If DII location descriptor is not included, then only the DII that signals the module that contains the ServiceGateway shall be taken into account when looking for modules matching a particular label.

The DII identifications in the loop should be sorted on importance. The DII that contains the label(s) with the highest pre-fetch priority should be listed first. Receivers that implement module-based pre-fetching should examine the DIIs for labels in the order in which they are listed in the DII location descriptor.

**Table 100 : Syntax of the DII location descriptor**

Syntax	Bits	Mnemonic
<code>DII_location_descriptor () {</code>		
<code>descriptor_tag</code>	8	uimsbf
<code>descriptor_length</code>	8	uimsbf
<code>transport_protocol_label</code>	8	uimsbf
<code>for(i=0; i&lt;N; i++ ) {</code>		
<code>reserved_future_use</code>	1	bslbf
<code>DII_identification</code>	15	uimsbf
<code>association_tag</code>	16	uimsbf
<code>}</code>		
<code>}</code>		

**descriptor\_tag:** This 8-bit field with value `0x0D` identifies this descriptor.

**descriptor\_length:** This 8-bit field identifies the number of bytes following the descriptor length field.

**transport\_protocol\_label:** This 8-bit field identifies the [Transport protocol descriptor](#) that specifies the transport connection that delivers the modules to which this prefetch descriptor refers. See "[transport\\_protocol\\_label](#)" on [page 220](#).

**DII\_identification:** This 15-bit field identifies the DII message. It corresponds to the identification portion of the `transactionId`. See [B.32, "Sub-fields of the transactionId" on page 409](#).

**association\_tag:** This 16-bit field identifies the connection (i.e. elementary stream) on which the DII message is broadcast.

## 10.9 DVB-J specific descriptors

### 10.9.1 DVB-J application descriptor

One instance of this descriptor shall be contained in the "application" (inner) descriptor loop of the AIT for each DVB-J application. It provides start-up parameter information.

**Table 101 : DVB-J application descriptor syntax**

	No.of Bits	Identifier	Value
<code>dvb_j_application_descriptor() {</code>			
<code>descriptor_tag</code>	8	uimsbf	
<code>descriptor_length</code>	8	uimsbf	
<code>for(i=0; i&lt;N; i++) {</code>			
<code>parameter_length</code>	8	uimsbf	
<code>for(j=0; j&lt;parameter_length; j++) {</code>			
<code>parameter_byte</code>	8	uimsbf	
<code>}</code>			
<code>}</code>			
<code>}</code>			

**descriptor\_tag:** This 8 bit integer with value `0x03` identifies this descriptor.

**parameter\_length:** This 8 bit integer specifies the number of bytes in the `parameter_byte` string.

**parameter\_byte:** The parameter bytes contain an array of strings that are passed to the application as parameters.

### 10.9.2 DVB-J application location descriptor

One instance of this descriptor shall be contained in the "application" (inner) descriptor loop of the AIT for each DVB-J application. It provides various items of path information to allow the DVB-J application to be found and then operated.

**Table 102 : DVB-J application location descriptor syntax**

	No.of Bits	Identifier	Value
<code>dvb_j_application_location_descriptor {</code>			
<code>descriptor_tag</code>	8	uimsbf	
<code>descriptor_length</code>	8	uimsbf	
<code>base_directory_length</code>	8	uimsbf	
<code>for(i=0; i&lt;N; i++) {</code>			
<code>base_directory_byte</code>	8	uimsbf	
<code>}</code>			
<code>classpath_extension_length</code>	8	uimsbf	
<code>for(i=0; i&lt;N; i++) {</code>			
<code>classpath_extension_byte</code>	8	uimsbf	
<code>}</code>			
<code>for(i=0; i&lt;N; i++) {</code>			
<code>initial_class_byte</code>	8	uimsbf	
<code>}</code>			
<code>}</code>			

**descriptor\_tag:** This 8 bit integer with value `0x04` identifies this descriptor.

**base\_directory\_length:** This 8 bit integer specifies the number of bytes in the `base_directory_byte` string.

**base\_directory\_byte:** These bytes contain a string specifying a directory name starting from the root of the file system with directories delimited by the slash character "/" (0x2F). This directory is used as a base directory for relative path names. This base directory is automatically considered to form the first directory in the class path (after the path to the system's classes).

If the base directory is the root the string shall be "/".

**classpath\_extension\_length:** This 8 bit integer specifies the number of bytes in the [classpath\\_extension\\_byte](#) string.

**classpath\_extension\_byte:** These bytes contain a string specifying a further extension for the DVB-J class path where the classes of the application are searched in addition to the base directory. The class path extension string contains path names where the elements in the path are delimited by the semicolon character ";" (0x3B). The elements of the path may be either absolute paths starting from the root of the file system or they can be relative to the base directory. The directories are delimited by the slash character "/" (0x2F) and absolute path names begin with the slash character "/" (0x2F).

**initial\_class\_byte:** These bytes contain a string specifying the name of the object in the file system that is the class implementing the Xlet interface.

This string is a DVB-J class name that is found in the class path (e.g. "com.broadcaster.appA.MainClass").

## 10.10 DVB-HTML Specific descriptors

### 10.10.1 DVB-HTML application descriptor

One instance of this descriptor shall be contained in the "application" (inner) descriptor loop of the AIT for each DVB-HTML application. It indicates the value of the application parameters and signals the control applied by the broadcaster on the state of the application.

**Table 103 : DVB-HTML application descriptor syntax**

	No. of Bits	Identifier	Value
<code>dvb_html_application_descriptor() {</code>			
<code>descriptor_tag</code>	8	uimsbf	N1
<code>descriptor_length</code>	8	uimsbf	
<code>appid_set_length</code>	8	uimsbf	
<code>for(i=0; i&lt;N1; i++) {</code>			
<code>application_id</code>	16	bslbf	
<code>}</code>			
<code>for(j=0; j&lt;N; j++) {</code>			
<code>parameter_bytes</code>	8	uimsbf	
<code>}</code>			
<code>}</code>			

**descriptor\_tag:** This 8 bit integer with value 0x08 identifies this descriptor.

**appid\_set\_length:** This 8 bit integer specifies the length of the list of [application\\_ids](#).

**application\_id:** The values of these 16 bit fields form a set of application ids (see 10.5, "Application identification" on page 214). This set is the set of application ids that are allowed to be associated with inner applications of a DVB-HTML application, see 8.9, "Xlet integration" on page 120.

**parameter\_bytes:** The parameter bytes contain the string that is appended to the application initial path as parameters.

## 10.10.2 DVB-HTML application location descriptor

This descriptor is for use in the application loop of the AIT. It indicates the physical location of the application entry point in the transport media.

**Table 104 : DVB-HTML application location descriptor syntax**

	No. of Bits	Identifier	Value
<pre> dvb_html_application_location_descriptor () {     descriptor_tag     descriptor_length     physical_root_length     for(i=0; i&lt;N1; i++) {         physical_root_bytes     }     for(i=0; i&lt;N; i++) {         initial_path_bytes     } } </pre>	8	uimsbf	N1
descriptor_length	8	uimsbf	
physical_root_length	8	uimsbf	
physical_root_bytes		uimsbf	
initial_path_bytes	8	uimsbf	

**descriptor\_tag:** This 8 bit integer with value `0x09` identifies this descriptor.

**physical\_root\_length:** This 8 bit integer specifies the length of the `physical_root_byte` string.

**physical\_root\_bytes:** These bytes contain a string specifying the physical root of the application entry point. The semantic of this string is transport protocol specific as shown in table 105 "Transport specific semantic of physical root bytes".

**Table 105 : Transport specific semantic of physical root bytes**

protocol_id	semantic
0x0000	reserved
0x0001	A directory specification
0x0002	One of the base URLs defined in the <a href="#">Transport protocol descriptor</a> signalled for the application (see 10.8.1.2, "Transport via IP" on page 225).
0x0003...0xFFFF	TBD

**initial\_path\_bytes:** These bytes contain a string specifying the URL path component to the entry point document. This path is relative to the root defined in the `physical_root_bytes` field.

### 10.10.2.1 Example

The following example describes the usage of the DVB-HTML application location descriptor.

An application author designs an HTML application in the following manner:

- The application data is distributed among several directories, let say an "image" directory and a "main" directory.
- The application entry point is an HTML document called "index.htm" and stored in the "main" directory.

### 10.10.2.2 Application Entry Point

From the application author's point of view, the application entry point is specified by the path "main/index.htm". This path is stored in the `initial_path_bytes` string of the location descriptor.

If the broadcaster inserts this application in a file system sub-directory called "application", the `physical_root_bytes` content of the location descriptor will be the string "application/".

If the broadcaster uses a transport via IP for this application, they shall signal the used protocol and IP address in the [Transport protocol descriptor](#) associated with this application and the [physical\\_root\\_bytes](#) field shall contain the corresponding URL string.

### 10.10.3 DVB-HTML application boundary descriptor

This descriptor is defined for use in the application loop of the AIT. It provides a regular expression that describes the data elements that form the application.

This descriptor is optional. When absent, the application boundary defaults to the complete set of all content coming from the transport signalled in the [Transport protocol descriptor](#) associated with the application.

Multiple boundary descriptors can be used for the same application. In this case, the equivalent global regular expression is the OR combination (union) of the individual regular expressions.

**Table 106 : DVB-HTML application boundary descriptor syntax**

	No. of Bits	Identifier	Value
<pre> dvb_html_application_boundary_descriptor {   descriptor_tag   descriptor_length   label_length   for(i=0; i&lt;N1; i++) {     label_bytes   }   for(i=0; i&lt;N; i++) {     regular_expression_bytes   } } </pre>	8	uimsbf	N1
descriptor_tag	8	uimsbf	
descriptor_length	8	uimsbf	
label_length	8	uimsbf	
label_bytes	8	uimsbf	
regular_expression_bytes	8	uimsbf	

**descriptor\_tag:** This 8 bit integer with value `0x0A` identifies this descriptor.

**label\_length:** This 8 bit integer specifies the length of the [label\\_bytes](#) string.

**label\_bytes:** These bytes contain a string specifying the label that is associated with the set of data identified by the regular expression. This label can be used for pre-fetching in a transport specific manner.

**regular\_expression\_bytes:** These bytes contain a string specifying the regular expression that can generate all URLs that are in the domain of the application.

See 9.3.1.4.1, "Regular Expression Syntax" on page 194.

## 10.11 Constant values

Table 107 : Registry of constant values (Sheet 1 of 2)

Where used	Type	Value	Where Defined	Scope
private data specifier descriptor	descriptor tag	0x5F	PSI & SI tables	SI
Data broadcast id descriptor		0x66	PMT	
Application Signalling Descriptor		0x6F	PMT	
Service identifier descriptor		0x71	SDT	
Label descriptor	descriptor tag	0x70	DII moduleInfo userInfo	SI-DAT
Caching priority descriptor		0x71		
Content type descriptor		0x72	BIOP objectInfo (note 1)	
reserved to MHP for future OC descriptors		0x73-0x7F	OC	
reserved to MHP for future use	table ID on AIT PID	0x00-0x73		MHP
Application Information Table		0x74		
reserved to MHP for future use		0x75-0x7F		
reserved for private use		0x80-0xFF		
Application descriptor	descriptor tag	0x00	AIT	MHP
Application name descriptor		0x01		
Transport protocol descriptor		0x02		
DVB-J application descriptor		0x03		
DVB-J application location descriptor		0x04		
External application authorisation descriptor		0x05		
Routing Descriptor IPv4		0x06		
Routing Descriptor IPv6		0x07		
DVB-HTML application descriptor		0x08		
DVB-HTML application location descriptor		0x09		
DVB-HTML application boundary descriptor		0x0A		
Application icons descriptor		0x0B		
Pre-fetch descriptor		0x0C		
DII location descriptor		0x0D		
delegated application descriptor		0x0E		
Plug-in descriptor		0x0F		
Application storage descriptor		0x10		
reserved to MHP for future use		0x11-0x5E		
private data specifier descriptor (note 2)		0x5F		
reserved to MHP for future		0x60-0x7F		
User defined (note 3)	0x80-0xFE			
MHP Object Carousel	data broadcast id	0x00F0	PMT, AIT	SI
reserved for MHP Multi Protocol Encapsulation		0x00F1		
reserved to MHP use		0x00F0 - 0x00FE	PMT, AIT	SI

**Table 107 : Registry of constant values (Sheet 2 of 2)**

Where used	Type	Value	Where Defined	Scope
MHP Application Service	service type	0x10	SDT	SI
NOTE 1: Strictly MessageSubHeader::ObjectInfo in the file message and the bound object info in a file binding of a directory or service gateway message.				
NOTE 2: The DVB SI private data specifier descriptor is defined for use in the <a href="#">Application Information Table</a> to introduce private descriptors.				
NOTE 3: All user defined descriptors shall be within the scope of a private data specifier descriptor (see <a href="#">10.4.7, "Use of private descriptors in the AIT" on page 213</a> ).				

## 10.12 Service Information

### 10.12.1 Service identifier descriptor

Zero or more service identifier descriptors may be included in the SDT description of a service. Each such descriptor defines a single textual identifier for the service. The syntax of this identifier is specified in [14.9.1, "Syntax of the textual service identifier" on page 359](#).

A single service identifier can be assigned to services in different physical networks even if they have different `original_network_id` and `service_id`. A given service identifier shall only be associated with services that are considered to be the same service.

NOTE: It is up to the service provider to decide which services are "same" and which are not. For example, two services in two different networks where the service have the same programme content but different regional adverts could be generally considered to be the "same" service. However, this decision is entirely up to the service provider.

More than one service identifier may be allocated to a service instance.

**Table 108 : Service identifier descriptor**

	No.of Bits	Identifier	Value
<code>service_identifier_descriptor () {</code>			
<code>descriptor_tag</code>	8	uimsbf	
<code>descriptor_length</code>	8	uimsbf	
for (i = 0; i < descriptor_length; i++) {			
<code>textual_service_identifier_bytes</code>	8	uimsbf	
}			
}			

**descriptor\_tag:** This 8 bit integer with value [0x71](#) identifies this descriptor.

**textual\_service\_identifier\_bytes:** These bytes contain the unique identifier for a service encoded using the normal encoding for text strings in DVB SI.

## 10.13 Plug-in signalling

Two signalling scenarios are defined for delegated applications and plug-ins:

- [Native signalling scenario](#)
- [MHP signalling scenario](#)

### 10.13.1 Native signalling scenario

In this scenario it is sufficient for the plug-in to be signalled as a normal MHP application. Optionally it could also be signalled as a plug-in using the [Plug-in descriptor](#) (see [10.13.4 on page 237](#)).

### 10.13.2 MHP signalling scenario

In this scenario MHP plug-in signalling can replace some or all of the native signalling of the delegated application. The MHP signalling allows the MHP terminal to:

- identify that one (or more) delegated applications are available
- the plug-in application that is required
- how to start the plug-in application
- how to introduce the delegated application to the plug-in application

The plug-in may use native signalling defined for the delegated application to locate, load and execute it. This is outside the scope of this specification.

Each delegated application is associated with an application descriptor (see [10.7.3, "Application descriptor" on page 219](#)). The semantics of this descriptor are maintained with the following comments:

- the profile and version information is specific to the application format and are not registered by the MHP. However, the MHP rules for comparing profile and version values are maintained. So, the values in these fields are an encoding of the application's own profile and version numbering scheme to ensure that they are compatible with the MHP rules.
- the transport protocol label field which is allowed to be empty if the delegated application is not transported by an MHP supported transport protocol.

The additional MHP signalling provided to support plug-ins is:

- ["delegated application descriptor"](#) (see [10.13.3 on page 236](#))

An optional descriptor for delegated applications.

- ["Plug-in descriptor"](#) (see [10.13.4 on page 237](#))

A mandatory descriptor for plug-in application using MHP signalling.

### 10.13.3 delegated application descriptor

Zero or one instance of this descriptor can be placed in either the common or application loops of the AIT of the delegated application.

This optional descriptor allows one or more specific plug-in applications to be identified for this delegated application. This overrides the default identification of the plug-in application based on the application type signalled in the AIT for the delegated application. If this descriptor is absent then no specific plug-in application is identified.



Where more than one application is identified they are listed in order from most preferred to least preferred. If none of the preferred applications is available to the MHP terminal then it shall use its default mechanism to identify the plug-in application for this delegated application.

**Table 109 : Plug-in reference descriptor**

	No.of Bits	Identifier
delegated_application_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for( i=0; i<N; i++ ){		
application_identifier	48	uimsbf
}		
}		

**descriptor\_tag:** This 8 bit integer with value [0x0E](#) identifies this descriptor.

**descriptor\_length:** This 8 bit field indicates the number of bytes following the descriptor length field.

**application\_identifier:** This 48 bit field identifies a specific application that can consume this application.

### 10.13.4 Plug-in descriptor

One or more plug-in application descriptors shall be placed in the application loop of each plug-in application.

This descriptor defines for an application that implements a plug-in the set of delegated application formats that it supports. The descriptor identifies both the type of delegated application supported and the set of profiles and versions supported. The normal MHP rules for matching an application to a compatible platform apply to when matching an delegated application with a suitable plug-in.

DVB-J applications signalled with this descriptor shall implement the API defined in [11.7.8, "Plug-in APIs" on page 267](#).

The plug-in shall be considered suitable for running content if the application\_type matches, and the version numbers are compatible according to the algorithm specified in [10.7.3, "Application descriptor" on page 219](#).

**Table 110 : Plug-in application descriptor**

	No.of Bits	Identifier
plugin_application_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
application_type	16	uimsbf
for( i=0; i<N; i++ ){		
application_profile	16	uimsbf
version.major	8	uimsbf
version.minor	8	uimsbf
version.micro	8	uimsbf
}		
}		

**descriptor\_tag:** This 8 bit integer with value [0x0F](#) identifies this descriptor.

**descriptor\_length:** This 8 bit field indicates the number of bytes following the descriptor length field.

**application\_type:** This 16 bit field identifies an delegated application type that this application can consume.

**application\_profile:** This 16 bit field is an integer value which represents the application type specific profile.

**version.major:** This 8 bit field carries the numeric value of the major sub-field of the profile version number.

**version.minor:** This 8 bit field carries the numeric value of the minor sub-field of the profile version number.

**version.micro:** This 8 bit field carries the numeric value of the micro sub-field of the profile version number.

## 10.14 Stored applications

### 10.14.1 Use of signalling defined in MHP 1.0

Storable applications follow the standard MHP 1.0 application signalling. An MHP terminal that doesn't provide storage or doesn't recognise these extensions will perceive storable applications as viable applications that potentially can be run from the broadcast connection.

So, the signalling described here augments the signalling described in MHP 1.0.

#### 10.14.1.1 Stored broadcast service related applications

For stored broadcast service related applications the broadcast AIT information shall be used when launching the stored application. So, little information from the AIT needs to be stored with the application.

#### 10.14.1.2 Stored stand-alone applications

For stored stand-alone applications, the AIT information used when launching the application shall come from a stored representation of the AIT. While a stored service is being presented in a service context, the following apply:

- The applications database shall be populated with information from this stored representation of the AIT for the applications forming part of that stored service
- The implementation shall report any changes in the stored representation of the AIT in the same way as it would report changes in a broadcast AIT

### 10.14.2 Application storage descriptor

This application storage descriptor advertises that an application can be stored and provides some indications of its properties. The presence of this application storage descriptor indicates that an "[Application description file](#)" is provided for the application (see [10.14.3 on page 239](#)). For a storable application a single application storage descriptor shall be placed in either the common descriptor loop or application descriptor loop of the AIT.

This descriptor, and the implied "[Application description file](#)", also supports receivers that implement speculative caching.

**Table 111 : Syntax of application storage descriptor**

	No.of Bits	Identifier
<code>application_storage_descriptor() {</code>		
<code>descriptor_tag</code>	8	uimsbf
<code>descriptor_length</code>	8	uimsbf
<code>storage_property</code>	8	uimsbf
<code>not_launchable_from_broadcast</code>	1	bslbf
<code>reserved</code>	7	bslbf
<code>version</code>	32	uimsbf
<code>priority</code>	8	uimsbf
<code>}</code>		

**descriptor\_tag:** This 8 bit integer with value `0x10` identifies this descriptor.

**descriptor\_length:** This 8 bit field indicates the number of bytes following the descriptor length field.

**storage\_property:** This 8 bit field indicates the characteristics of the application w.r.t storage.

**Table 112 : Semantics of storage property values**

storage_property	property
0	broadcast related
1	stand alone
2 to 255	reserved

If the storage property is "broadcast related" then the application's life cycle is controlled by the broadcast service. Such applications are suitable for caching but not for running as a stand alone application.

If the storage property is "stand alone" then the application can usefully be launched by the user independently of a broadcast service. Applications with this property may also be launched as if broadcast related if the currently selected service lists them in its AIT.

**not\_launchable\_from\_broadcast:** This 1 bit field is a flag. When set to "1" this indicates that the delivery characteristics of this application are such that it is not useful to launch the application unless it has been completely cached/stored. If set to '0' then caching provides some benefit but is not essential.

Applications in stored services and applications where the **not\_launchable\_from\_broadcast** bit is set to "1" shall only be launched where the MHP terminal has stored the complete set of files which are listed in the "[Application description file](#)" as being critical.

**version:** This 32 bit field provides the version number of the application. This number starts at zero and increments by one each time any of the files listed in the "[Application description file](#)" change or the contents of the "[Application description file](#)" itself changes. Used values shall never reused, in the event that the number range is exhausted a new application id shall be used.

**priority:** This field indicates the priority of this application for storage relative to the other applications signalled in this service.

It is only meaningful for applications which have been proactively cached by the MHP terminal implementation and shall be ignored otherwise.

Higher values indicate more important applications to store. The behaviour when applications have the same priority is implementation dependent.

## 10.14.3 Application description file

### 10.14.3.1 Description

The "[Application description file](#)" provides the list of files that need to be installed as well as other related necessary information. The notation uses an XML-based syntax.

For those applications that can be stored, an "[Application description file](#)" file shall be placed in the same carousel as the application.

It should be noted that the application description file does not duplicate all the information needed to run the application, the MHP terminal needs to use also the information in the AIT when installing the application.

Receivers that don't have the complete set of files listed by the "[Application description file](#)" cached/stored should not attempt to launch this application.

Where a file is listed in the "[Application description file](#)" of more than one application and is stored, the MHP terminal shall ensure that each application sees the correct version of the file for that application. The version of the file visible to one application shall not be changed by any changes in the version of the file visible to any other application which may share that same file.

### 10.14.3.2 Application description file name and location

The application description file shall be located in the base directory of a DVB-J application (as signalled in the dvb-j application location descriptor) or the physical root of a DVB-HTML application (as signalled in the DVB-HTML application location descriptor). By convention, the name of a ADF is:

```
'dvb.storage.oooooooo.aaaa'
```

where:

oooooooo is the organisation id of the application as a 8 character hexadecimal string

aaaa is the application id as a 4 character hexadecimal string

See also 10.14.2, "Application storage descriptor" on page 238.

### 10.14.3.3 Syntax

The syntax of the "Application description file" is defined by the following XML DTD.

The `PublicLiteral` to be used for specifying this DTD in document type declarations of the XML files is:

```
"-//DVB//DTD Application Description File 1.0//EN"
```

and the URL for the `SystemLiteral` is:

```
"http://www.dvb.org/mhp/dtd/applicationdescriptionfile-1-0.dtd"
```

```
<!-- the main element for the application description -->
<!ELEMENT applicationdescription (object+)>

<!ENTITY % object "(dir|file)">

<!ELEMENT dir (%object;)*>
<!ATTLIST dir
  name CDATA #REQUIRED
  priority NMTOKEN #IMPLIED
>

<!ELEMENT file EMPTY>
<!ATTLIST file
  name CDATA #REQUIRED
  priority NMTOKEN #IMPLIED
  size NMTOKEN #REQUIRED
>
```

### 10.14.3.4 Semantics

**name:** This attribute provides the name of a file system object (directory or file) that is storable. This is the name of the object within its enclosing directory and hence does not include any directory path information.

**NOTE:** No elements are provided for naming object types such as Stream or StreamEvent therefore there is no mechanism to specify that Stream and StreamEvent objects are required to be stored.

**priority:** This attribute describes how important it is to store this object. The value zero indicates that it is critical to store the object (i.e. there is no benefit in storing any objects unless this part is stored). Higher values indicate lower storage priority.

The default value for the priority attribute is zero (i.e. critical).

The `priority` for an object inherits from the immediately enclosing directory.

**size:** This attribute defines the size in bytes of the file.

---

# 11 DVB-J Platform

## 11.1 The Virtual Machine

The DVB-J virtual machine is defined in [Java VM \[34\]](#), as amended by [JVM Errata \[68\]](#) plus the Inner Classes specification in [Inner Classes \[69\]](#).

The Java Virtual Machine shall support Java class files whose version number is in the range 45.3 through 45.65535.

## 11.2 General issues

### 11.2.1 Basic Considerations

Unless otherwise specified in this specification, MHP implementations are not required to include any classes or methods marked as deprecated in those API specifications which are either referenced by or included in this specification.

A DVB-J application is considered to logically run in its own virtual machine instance. For this reason, it cannot rely on finalizers that are defined in application classes being run when the application terminates. When the application manager terminates the entity that represents the virtual machine in which the application is run, a conformant implementation is permitted to not run application finalizers, as spelled out in section 2.17.9 of the [Java Language Spec \[32\]](#).

DVB-J applications shall not synchronize on system classes or other exposed system static objects else undefined behaviour may occur.

SecurityExceptions shall only be thrown either where they are declared as part of the description of the method concerned or where identified in this specification for those referenced packages which do not include any SecurityExceptions.

It is an allowable implementation choice to not override methods as long as the defined semantics are respected. The implication of this is that such differences between implementations will be visible when using the `java.lang.reflection` package.

The inclusion of java packages shall not imply the inclusion of other sub packages e.g. the inclusion of the `javax.tv.media` package does not automatically imply the inclusion of the `javax.tv.media.protocol` sub package. Inclusion of sub packages shall be explicitly listed in this specification.

DVB-J applications shall not use additional public or protected methods or fields in the `org.dvb` namespace which are not listed in this specification. Applications which scale to run on multiple revisions of this specification shall only use `org.dvb` methods etc. appropriate to the version of the platform on which they are running.

### 11.2.2 Approach to Subsetting

Where a class included in this specification has methods with signature dependencies on classes not included in the implemented profile, these methods are not required to be present in an implementation. A compliant implementation choice may require these methods and classes to be present.

Where this specification subsets a package, inclusion of the complete package is allowed but clearly not required. The behaviour of the additional features is not specified for broadcast applications.

### 11.2.3 Class Loading

The DVB-J application environment shall be written such that each application appears to run within its own classloader or classloader hierarchy for all classes that are not a part of the platform. As a consequence, two applications will never be able to access the same copy of any application-defined static variable.

Signed applications shall only load classes signed by the same leaf certificate.

See 12, "Security" on page 286.

Inappropriate, meaningless or illegal locators are silently skipped and searching continues.

## 11.2.4 Unloading

Class unloading as defined by section 12.8 of [Java Language Spec \[32\]](#) and section 2.16.8 of [Java VM \[34\]](#) will be supported.

## 11.2.5 Event listeners

In `org.dvb` and `org.davic` all methods to remove event listeners shall have no effect if the listener is not registered.

## 11.2.6 Event model in DAVIC APIs

Events defined in the DAVIC APIs [DAVIC 1.4.1p9 \[3\]](#) shall extend `java.util.EventObject`. Event listeners defined in [DAVIC 1.4.1p9 \[3\]](#) shall extend `java.util.EventListener`.

## 11.2.7 Event model in DAVIC & DVB APIs

Each class in `org.dvb` and `org.davic` inheriting from `java.util.EventObject` is just a container for these fields and no validity checks are done for the parameters by this constructor. Instances of these classes are intended to be constructed by the platform implementation and not by applications. The platform implementation will only construct these events with the appropriate information passed in.

In `org.dvb` and `org.davic` all methods to add event listeners add each listener only once if the add method is called with the same parameters multiple times. This means that the same event is delivered only once to each listener even if it has been added twice.

## 11.2.8 Tuning as a side-effect

No MHP API shall cause tuning unless explicitly specified as such. For example, if a locator that requires tuning is received by the `DVBClassLoader` it shall behave as if the specified file is not available.

## 11.2.9 Intra application media resource management

Where an application makes conflicting requests for limited media decoding resources, the media decoding resources that are requested most recently are presumed to be the ones that are most wanted. This applies between MPEG-2 I-Frames, MPEG-2 Video "drips" and streaming video. Similarly, this applies between streaming audio and audio from memory.

When a non broadcast media presentation (audio or video from memory or still image) is interrupted by a resource loss within the same application, the first presentation is cancelled and will not be restored.

If a broadcast media presentation is interrupted by a resource loss within the same application, the broadcast presentation is restored when the interrupting presentation ends.

## 11.2.10 Application thread priority

The `ThreadGroup` of application threads shall have a `MaxPriority` of 5.

NOTE: As a consequence applications will not be able to create threads at priorities greater than 5 since they don't have `java.lang.RuntimePermission("modifyThread")`. Applications may perform compute intensive tasks within application created threads without being considered unresponsive.

## 11.3 Fundamental DVB-J APIs

### 11.3.1 Java platform APIs

The following packages are defined in [JAE 1.1.8 API \[31\]](#). See also 11.8.1.3, "Other classes" on page 269.

#### 11.3.1.1 java.lang package

The java.lang package is supported with the following modifications.

- a) The following methods shall always throw a SecurityException when called by inter-operable applications:

```
Runtime.exec(),
Runtime.load(),
Runtime.loadLibrary(),
System.exit(),
System.load(),
System.loadLibrary(),
```

- b) MHP implementations are not required to include the following methods:

```
Runtime.runFinalizersOnExit(),
System.runFinalizersOnExit(),

Thread.destroy(),
Thread.stop(),
Thread.suspend(),
Thread.resume(),
Thread.countStackFrames(),
ThreadGroup.stop(),
ThreadGroup.suspend(),
ThreadGroup.resume(),
ThreadGroup.allowThreadSuspension()
```

- c) The following fields shall not be used by inter-operable applications:

```
System.in
```

- d) The following classes shall not be used by inter-operable applications:

```
java.lang.Process
```

- e) Applications shall be able to use:

```
System.out,
System.err,
Runtime.traceInstructions(),
Runtime.traceMethodCalls()
```

for debugging without any adverse effects to the application. The output shall not be visible to normal end users and shall not conflict with any other API.

- f) The `java.lang.Compiler` class and following methods shall be taken as hints from an application to the system however there is no guarantee of what happens:

```
Runtime.gc(),
System.gc()
```

g) Only the following properties are required to be supported for `System.getProperty()` and `System.getProperties()`:

`file.separator`,  
`path.separator`,  
`line.separator`,

`dvb.persistent.root` (see section 11.5.6, "Persistent Storage API" on page 258)

With the exception of `dvb.persistent.root` all of these properties are accessible to unauthorised applications. `dvb.persistent.root` is accessible only to authorised applications.

Property names beginning "dvb." are reserved for future use.

h) The `System.setProperties()` and `System.setSecurityManager()` methods will always throw an exception when called by downloaded DVB-J applications.

i) The class `SecurityManager` shall be as specified in [PersonalJAE \[36\]](#).

The call `System.currentTimeMillis()` shall feature a granularity of the returned time value of not more than 10 ms. With the call `Object.wait(long timeout)`, the timeout value is specified as a maximum time to wait. The MHP further guarantees that if not notified the object will wait for at least `timeout - 10` ms.

### 11.3.1.2 `java.lang.reflect` package

The `java.lang.reflect` package is supported.

### 11.3.1.3 `java.util`

The `java.util` package is supported. The constants in the `Locale` class do not imply support (or otherwise) for these Locales. Locales supported in the MHP are specified in profiles (see 15.4, "Locale support" on page 364).

### 11.3.1.4 `java.util.zip`

The `java.util.zip` package is supported with the exception of the following classes:

- `Deflater`
- `DeflaterOutputStream`
- `GZIPOutputStream`
- `ZipOutputStream`
- `Adler32`

### 11.3.1.5 `java.io`

The `java.io` package is supported.

### 11.3.1.6 `java.net`

From the `java.net` package, the `java.net.URL` and `java.net.InetAddress` classes and the exceptions `MalformedURLException` and `UnknownHostException` only are supported. Unless included in the platform as part of a specific profile, All the rest of this package is to be non-supported as defined in section 11.2.2, "Approach to Subsetting" on page 241. The signature dependencies from these to the rest of this package are severed as described in 11.2.2, "Approach to Subsetting" on page 241.

Support is required for at least one implementation dependant `java.net` URL protocol. The `Class.getResource` method shall return instances of `java.net.URL` using such a protocol where no appropriate protocol is defined in this specification.

Support is required for the "file:" protocol. Applications shall be able to construct instances of `java.net.URL` using strings containing "file:" URLs as defined in [RFC 1738 \[67\]](#) where the `<host>` element is the empty string and the `<path>` element is an absolute filename.



The method `URL.getContent` shall work as specified in its specification even though the reference to the URL connection is not valid in all MHP profiles.

The return types of `URL.getContent()` are defined by the mappings from data type to java class name listed in table 113 "Return types of `URL.getContent()`".

**Table 113 : Return types of `URL.getContent()`**

Data type	Return type
Unknown or unsupported data types	<code>java.io.InputStream</code>
<code>text/plain</code>	<code>java.io.InputStream</code>
<code>text/dvb.utf8</code>	<code>java.io.InputStream</code>
<code>text/Generic</code>	<code>java.io.InputStream</code>
<code>image/png</code>	<code>java.awt.image.ImageProducer</code>
<code>image/jpeg</code>	<code>java.awt.image.ImageProducer</code>
<code>image/mpeg</code>	<code>org.havi.ui.HBackgroundImage</code>

The behaviour of `URL.getContent()` responds to data type using information with the following priority:

- a) Content type signalling such as:
  - the content type descriptor in the OC (see B.2.3.4, "Content type descriptor" on page 394).
  - the HTTP header (if supported in the profile) the Content-Type header field (if present)
- b) The filename extension (if known) (see table 6, "File type identification" on page 70)
- c) Open the file & study

`getContenttype` returns the value contained in the optional content type descriptor in the OC (see B.2.3.4, "Content type descriptor" on page 394) if present.

`getFileNameMap` returns information derived from table 6, "File type identification" on page 70.

### 11.3.1.7 `java.beans`

The `java.beans` package is supported.

### 11.3.1.8 `java.math`

From the `java.math` package `java.math.BigInteger` is supported.

### 11.3.1.9 `java.text`

The `java.text` package is supported with the exception of the following classes and interfaces:

- `CharacterIterator`
- `BreakIterator`
- `CollationElementIterator`
- `CollationKey`
- `Collator`
- `RuleBasedCollator`
- `StringCharacterIterator`

## 11.3.2 MHP platform APIs

### 11.3.2.1 org.dvb.lang

The `org.dvb.lang` package is supported as defined in annex H, "(normative): DVB-J fundamental classes" on page 449.

The valid types of locator for use with `DVBClassLoader` shall include the following:

- `org.davic.net.dvb.DVBLocator`

That contains a valid reference to a component within a service and file path component referencing a directory within the carousel identified by the referenced component.

Where no parent is specified at creation, the delegation parent classloader of `DVBClassLoader` shall be the original classloader of the calling application.

ClassLoader delegation is defined in the specification for `java.lang.ClassLoader` in [PersonalJAE \[36\]](#).

### 11.3.2.2 org.dvb.event

The `org.dvb.event` package is supported as defined in annex I, "(normative): DVB-J event API" on page 453.

When sending events from a `org.dvb.event.EventManager` to listeners, the implementation shall ensure that any single listener shall only be sent one platform generated event instance at one time. If a new event is generated before the listener method has returned from processing a previous event, the MHP terminal shall not call that listener method until the call for processing the previous event returns.

The behaviour where applications have multiple such event listeners registered is implementation dependent. Implementations may use multiple threads to send the same event instance to any such multiple listeners. Implementations may ensure that at most one event listener in any one application for all these events is executing at any one time.

## 11.4 Presentation APIs

### 11.4.1 Graphical User Interface API

#### 11.4.1.1 The Core GUI API

The following package is defined in [JAE 1.1.8 API \[31\]](#).

The following classes and interfaces from the `java.awt` package are included in the MHP specification:

- `Adjustable`
- `AWTError`
- `AWTEvent`
- `AWTEventMulticaster`
- `AWTException`
- `BorderLayout`
- `CardLayout`
- `Color`
- `Component`
- `Container`
- `Cursor`
- `Dimension`
- `EventQueue`
- `FlowLayout`
- `Font`
- `FontMetrics`
- `Graphics`
- `GridBagConstraints`
- `GridBagLayout`
- `GridLayout`
- `IllegalComponentStateException`
- `Image`
- `Insets`
- `ItemSelectable`
- `LayoutManager`
- `LayoutManager2`
- `MediaTracker`
- `MenuContainer`
- `Point`
- `Polygon`
- `Rectangle`
- `Shape`
- `Toolkit` (as detailed below)

The entire `java.awt.event` package is included.

Property names for use with the `getProperty` method on `java.awt.Image` and its sub classes beginning "dvb." are reserved for future use.

See also [11.8.1.3, "Other classes" on page 269](#).

The signature dependencies from `Component` to `PopupMenu` (the `add` method) and `MenuComponent` (the `remove` method) are severed as described in [11.2.2, "Approach to Subsetting" on page 241](#). The same applies for all references to the `java.awt.peer` package.

In the `Toolkit` class, the following methods are required:

- `getDefaultToolkit`,
- `getFontList`,
- `getFontMetrics`,
- `sync`,
- `getColorModel`,
- and all the methods relating to images.

The following two methods are added to the Toolkit class from [PersonalJAE \[36\]](#). These two methods shall not cache `java.awt.Image` objects.

- `public abstract Image createImage(String filename);`
- `public abstract Image createImage(URL url);`

The image caching behaviour of the following two methods is now observed as explicitly specified in [PersonalJAE \[36\]](#). These two methods cache `java.awt.Image` objects so that two calls with the same argument return the same object.

- `public abstract Image getImage(String filename);`
- `public abstract Image getImage(URL url);`

Applications shall be able to use `Toolkit.beep` without any adverse effects to the application. The output is not required to be audible to normal end users and shall not conflict with any other API.

The methods `getScreenResolution` and `getScreenSize` shall be supported with the additional semantics described in [HAVi \[50\]](#). The methods `getSystemEventQueue` and `getSystemEventQueueImpl` are supported however each Xlet will have a unique queue not shared with other Xlets or anything else in the system.

All of the `java.awt.image` package is required. The encoding of image content types for use by `java.awt.image` are defined in [7.1.1, "Bitmap image formats" on page 64](#). The set of formats supported is profile dependent.

When using the `java.awt.FontMetrics` class, the width of a set of characters or string returned by the `charsWidth` or `stringWidth` method shall be correct taking into account any kerning and sub-pixel positioning applied by the font renderer. Calculating the same number by adding the widths of the individual characters is not required or expected to return the same number since it will not take into account any kerning or sub-pixel positioning applied by the font renderer.

The downloading of fonts from the network (see [D.2.2, "Downloaded fonts" on page 421](#)) shall be supported using the methods concerned on `org.dvb.ui.FontFactory`. Failure to download a font shall be reported by these methods. The constructor for `java.awt.Font` shall only be aware of platform resident fonts.

#### 11.4.1.2 TV user interface

The packages `org.havi.ui` and `org.havi.ui.event` defined in [HAVi \[50\]](#) shall be supported. Instances of `org.havi.ui.event.HRCEvent` are reported through the normal `java.awt` event mechanism due to the inheritance from `java.awt.event.KeyEvent`.

With the exception of the `HSound.load` method, no methods specified in [HAVi \[50\]](#) shall throw a security exception in the MHP context. The permissions for `HSound.load` are those defined for `java.io`.

The `DVBTextLayoutManager` specified in [U, "\(normative\): Extended graphics APIs" on page 753](#) shall be supported.

The following semantics shall be used for the `getVideoController` method on `HVideoDevice`.

- It shall only return JMF players (see [Java Media Player Specification \[33\]](#)) which are in the Prefetched or Started states and which are using that `HVideoDevice` as one of their scarce resources. Otherwise null will be returned. It shall not return JMF players from other applications if those are using the video device underlying the `HVideoDevice`.
- Except as specified below, it shall only return JMF players which have been already created in response to the application calling `javax.media.Manager.createPlayer` or which have been returned by `javax.tv.service.selection.ServiceContext.getServiceContentHandlers` (see [Java TV \[51\]](#)).
- The only exception to the above is the situation where video is being played in the background as part of the context of an application but where `javax.tv.service.selection.ServiceContext.getServiceContentHandlers` has not yet been called. In this case, `getVideoController` shall return the same JMF player as would be returned by `getServiceContentHandlers` if it was to be called subsequently.

The signatures of the classes `HComponent` and `HContainer` shall be extended with `"implements org.dvb.ui.TestOpacity"`.

The methods `HGraphicsConfiguration.getPunchThroughToBackgroundColor` shall not be used by inter-operable applications.

The methods `fontAvailable()` and `downloadFont()` in the class `org.havi.ui.HFontCapabilities` shall not be used by inter-operable applications.

The package `javax.tv.graphics` defined in [Java TV \[51\]](#) shall be supported.

### 11.4.1.3 Extended graphics

See [U, "\(normative\): Extended graphics APIs" on page 753](#).

### 11.4.1.4 Handling of input events

The MHP includes two ways for applications to receive input events: the normal AWT method in `java.awt.Component` and the `org.dvb.event` package (see [11.3.2.2, "org.dvb.event" on page 246](#)). Via the normal AWT method, the application normally receives all input events when the component has the focus. The minimum set of input events that are supported is defined in [G.5, "Input events" on page 446](#).

Often the resident Navigator software of the MHP terminal uses many of the keys on the remote control for its own navigation purposes. An MHP application having the AWT focus will consume the defined input events by itself which possibly may cause them not to be delivered to the resident Navigator.

To ensure consistent user experience, the following rules are defined:

- an application creating a `Scene` and placing components into it shall not by default get the input focus for these components
- the application may request to get the input focus by calling `Component.requestFocus()`. If this is granted and the focus moved to the requested component, this component shall receive input events as defined in [1.1 on page 453](#).
- the application may request to receive a subset of input events via the `org.dvb.event` API even when not having the AWT focus.

For applications delivered within normal television services, it is recommended that the following items are taken into account:

- when the display is primarily showing the video of the television service and the user perception is that he is not actively interacting with an application but is just watching the television service (this state is later called 'television viewing mode'), the applications should not request the AWT focus but let most of the input events go to the resident Navigator (e.g. number keys, directional arrow keys and Enter may cause the normal actions the user expects from the Navigator)
- in the 'television viewing mode' the applications should request only the minimum required input events, e.g. only input events that cause some further action to happen and that may transition the user from the 'television viewing mode' into a state where the user more actively interacts with the applications.
- if the applications require some input events for the user to be able to activate the application when in the 'television viewing mode', it is recommended that the `VK_COLOURED_KEY_(0...3)` and `VK_TELETEXT` (see note below) input events are used for this purpose and the applications request them via the `org.dvb.event` API. In practice this means that the navigator can not map any essential function directly to the `VK_COLOURED_KEY(0...3)` events in television viewing mode.
- MHP applications wishing to leave "tv viewing mode" and receive events other than the `VK_COLOURED_KEY_(0...3)` and `VK_TELETEXT` are obliged to display a non-transparent graphics object on the screen that covers at least 3 % of the visible area on the screen (equivalent to 9852 pixels on a 634x518 visible screen in a 720x576 graphics mode) which clearly indicates to the user the "non-navigator" mode of the receiver with respect to input event handling.

NOTE 1: In television services where DVB Teletext is transmitted within the service, a typical Navigator action in the 'television viewing mode' can be to activate a teletext decoder, if supported by the terminal. The application requesting this input event may cause the DVB Teletext decoder not to be conveniently accessible to the end user. Therefore, within services that carry DVB Teletext it is recommended that the VK\_TELETEXT input event is requested in the 'television viewing mode' only by such applications that provide a simulcast MHP application version of the teletext service, thus providing a replacement for the DVB Teletext information.

In environments where WST Teletext services are prevalent when there is no DVB Teletext service associated with the video service, the MHP application should display an information service of a similar nature to teletext. When receiving this event for the second time the information service should terminate and the user will return to normal viewing mode.

The VK\_TELETEXT event should have no other purposes in MHP applications.

NOTE 2: The items above apply only to the 'television viewing mode' as explained above. In states where the user experience is that the user is clearly primarily interacting actively with an application the applications can naturally use the normal AWT input event mechanism and request the focus as well as use any of the supported input events via the org.dvb.event API.

NOTE 3: For data services, the first interaction state of an application that is automatically started after service selection should behave as television viewing mode. This is in order to allow users who are navigating with the navigator to seamlessly navigate through these services without getting confused.

## 11.4.2 Streamed Media API

### 11.4.2.1 Framework of solution

The `javax.media` and `javax.media.protocol` packages from Java Media Framework as defined in [Java Media Player Specification \[33\]](#) shall be implemented with the clarifications, extensions and restrictions as defined in the corresponding sections below.

### 11.4.2.2 Clarifications

The JMF "time base time" when playing MPEG content delivered in MPEG transport streams is used as a constantly progressing time whose rate may be synthesized from the Program Clock References / the System Time Clock in MPEG or by some other appropriate method. The value does not have any direct relation to the value of the MPEG System Time Clock in the receiver.

The media time of JMF when playing MPEG content delivered in MPEG transport streams is just a time value that progresses as the media stream is played, but whose actual value is implementation dependent.

NOTE: There is no implied or expected connection between the JMF media time and any time base(s) provided by the MPEG-2 / DSM-CC Normal Play Time.

When creating a JMF player, Locators and URLs which reference a DVB service, event or elementary stream will create a player which plays the content concerned direct from the network. Locators and URLs which reference files will create a player which will download the content concerned from the network during the Prefetching state of the player concerned. If the content cannot be downloaded then such players will never enter the Prefetched state.

Interoperable applications shall not call `javax.media.MediaHandler.setSource()`.

In the `javax.media.PackageManager`:

- a) The effect of the `set<xxx>PrefixList` methods is limited to the application calling the method.
- b) The `SecurityException` is always thrown in MHP.

`SubtitleLanguageControl.isSubtitlingOn()` returns the current state of subtitle presentation and NOT the last value set with `setSubtitling`. In particular `setSubtitling` may be set to true but if there are no subtitles in the network or no subtitle service component selected then `isSubtitlingOn` shall return false.

Any elements in `DVBLocators` after and including the `event_id` element are ignored by JMF players.

### 11.4.2.3 Default media player behaviour

For a JMF player which is presenting a DVB service, the following rules will be followed in the order given to decide which service components will be presented when multiple audio, video or subtitle components are present in a service:

- a) For audio and subtitle streams in different languages, user preferences will be used to determine which streams are selected.
- b) The streams which are first in the network signalling information (i.e. in the appropriate SI table) will be presented.

If any of the media components comprising the service change then the implementation will as far as possible replace the changed components with suitable replacements in a user preference and implementation dependant manner.

### 11.4.2.4 Required controls for video drips

The following controls are supported for video drips (see 7.1.3, "MPEG-2 Video "drips"" on page 64):

- `javax.tv.media.AWTVideoSizeControl`
- `org.dvb.media.BackgroundVideoPresentationControl`

### 11.4.2.5 Extensions to the Framework

#### 11.4.2.5.1 DVB specified extensions

The classes and interfaces defined in annex N, "(normative): Streamed Media API Extensions" on page 574 of this specification are included.

#### 11.4.2.5.2 Extensions in `org.davic`

The following classes and interfaces will be included from the `org.davic.media` package, as defined in annex L of [DAVIC 1.4.1p9 \[3\]](#).

- |  |  |
|--|--|
| • <code>MediaPresentedEvent</code>           | • <code>LanguageNotAvailableException</code> |
| • <code>MediaLocator</code>                  | • <code>NotAuthorizedException</code>        |
| • <code>MediaTimePositionControl</code>      | • <code>NotAuthorizedMediaException</code>   |
| • <code>FreezeControl</code>                 | • <code>MediaFreezeException</code>          |
| • <code>ResourceWithdrawnEvent</code>        | • <code>LanguageControl</code>               |
| • <code>ResourceReturnedEvent</code>         | • <code>AudioLanguageControl</code>          |
| • <code>MediaTimePositionChangedEvent</code> | • <code>SubtitlingLanguageControl</code>     |

The following classes will be included from the `org.davic.media` package as defined in annex L of [DAVIC 1.4.1p9 \[3\]](#) with the following semantic modification:

- The completion of the action started by calling `setMediaTimePosition()` on the `MediaTimePositionControl` will be signalled by a `org.davic.media.MediaTimePositionChangedEvent`.
- The `org.davic.media.MediaPresentedEvent` shall only be thrown following changes caused using controls in the `org.davic` package and as part of the transition of a JMF player into the Started state. In this second case, it augments the JMF state transition events and does not replace them.

NOTE: Changes effected using `javax.tv.media.MediaSelectControl` do not lead to the `org.davic.media.MediaPresentedEvent` being thrown.

### 11.4.2.5.3 Extensions in javax.tv

The package `javax.tv.media` defined in [Java TV \[51\]](#) shall be implemented.

In `javax.tv.media.MediaSelectControl` the `InsufficientResourcesException` is thrown by the `select()` method if selecting any service component fails due to a lack of system resources.

In `javax.tv.media.AWTVideoSize.equals()`, the "data members" shall be considered to be the "source" and "dest" parameters used to create the object.

### 11.4.2.5.4 Required controls for broadcast profiles

The following controls are supported for the broadcast streaming formats specified in 7.2, "[Broadcast streaming formats](#)" on page 66:

- `org.davic.media.LanguageControl`
- `org.davic.media.AudioLanguageControl`
- `org.davic.media.SubtitlingLanguageControl`
- `org.davic.media.FreezeControl`
- `javax.tv.media.MediaSelectControl`
- `javax.tv.media.AWTVideoSizeControl`
- `org.dvb.media.VideoPresentationControl`
- `org.dvb.media.BackgroundVideoPresentationControl`
- `org.dvb.media.SubtitlingEventControl`
- `org.dvb.media.VideoFormatControl`

The following controls are supported for media decoded from 7.1, "[Static formats](#)" on page 64:

- `org.davic.media.MediaTimePositionControl`

### 11.4.2.5.5 Clarifications

In `SubtitlingLanguageControl`, subtitling being presented shall be defined as the subtitle decoder running. It does not require subtitles to be visible on screen at that time. See 13.5, "[Subtitles](#)" on page 349.

When a `PresentationChangedEvent` is generated, the Player in question should re-evaluate the list of controls returned by the `getControls()` method for that player. Similarly, Players supporting the `MediaSelectControl` should re-evaluate the list of Controls returned by `getControls()` after calls to the `MediaSelectControl.select()` method is called.

Any controls referenced by an application after a `PresentationChangedEvent` or a `MediaSelectSucceededEvent` is generated may fail silently if they are no longer valid. Controls which are valid for the new content remain unaffected. Applications should re-acquire any controls they require after either of these events has been generated.

### 11.4.2.6 Restrictions on the Framework for Broadcast

Controls that are supported the MHP need not have an associated GUI component. Any calls to `Control.getComponent()` may return null.

Developers of MHP applications should not rely on the presence of the following classes or interfaces:

- `javax.media.CachingControl` (unless returned by a call to a JMF method)
- `javax.media.CachingControlEvent`
- `javax.media.GainControl` (unless returned by a call to a JMF method).



An MHP implementation need not return a `CachingControl` or `GainControl` from a call to `Player.getControls()` however this is not prohibited. If an MHP implementation does return a `GainControl`, then the volume that is set using this control may not be greater than the system volume level. I.e. the gain control may only change the volume between mute and the volume level at the time the application was started. This system volume level shall be represented as 1.0.

JMF Players are not required to return a `java.awt.Component` from their `getVisualComponent` method and may return null. Players which return null can only be used to present video in the background. Players which return a `java.awt.Component` must fully support the semantics for `java.awt.Component` concerning positioning and scaling.

Players start as background players. If a `getVisualComponent` returns a component then a player is also capable of performing as a component based player. Video shall continue running in the background until the first call to the paint method of the component returned `getVisualComponent`. When this transition happens the video is resized and repositioned as required by the `visualComponent`. Possible areas outside of the component are updated to follow the rules in 13.3, "Graphics" on page 338.

After this transition, applications shall re-acquire the list of JMF controls for the player. It is implementation dependent whether any previous JMF controls are re-used. Applications shall not use any previous JMF controls which are not in the new list of JMF controls. When a JMF player is performing as a component based player, the following JMF controls shall not be supported and video scaling & positioning shall be done by controlling the size & position of the component returned by the `getVisualComponent` method.

- `org.dvb.media.BackgroundVideoPresentationControl`
- `javax.tv.media.AWTVideoSizeControl`

Applications should not expect `PushDataSource` and `PullDataSource` to exist for broadcast MPEG content, and it is not required that functional implementations be provided for implementations of JMF in DVB. The `DataSource` used may be implementation-specific and non-specified apart from its inheritance from `javax.media.protocol.DataSource`.

The constructor for `URLDataSource` may always throw `IOException` for broadcast MPEG content.

For the purposes of integration with `javax.tv.media.MediaSelectControl` "resynchronization" is not considered to occur provided that the same `PCR_PID` is referenced between the new and existing service components.

Implementations of JMF shall not tune but shall allow selection of media components from transport streams which can be reached without tuning even where they are not part of a currently selected service.

#### 11.4.2.7 Intersection Between `MediaSelectControl` and `SubtitlingLanguageControl` / `AudioLanguageControl`

The method `org.davic.media.LanguageControl.listAvailableLanguages()` shall list all available languages in the service concerned. This includes languages carried in service components which are not currently selected, either because the service component concerned has been de-selected using `MediaSelectControl` or because it was never selected initially because it was not carried in one of the components listed in the component tags of a locator which included component tags.

The method `org.davic.media.LanguageControl.selectLanguage()` shall over-ride any selection of service components made either by `javax.tv.media.MediaSelectControl` or by using a locator for a service which included component tags. If `selectLanguage()` has changed the set of service components selected, the methods on `javax.tv.media.MediaSelectControl` shall behave as if that change had been made through `MediaSelectControl` and return the correct set of service components taking into account the change made by `selectLanguage()`.

If the methods on `javax.tv.media.MediaSelectControl` are used to replace or remove the service component carrying the currently active audio or subtitles then the presentation of this service component shall stop. If the methods on `javax.media.MediaSelectControl` are used to add a service component carrying audio or subtitles when a service component of this media type is not already being presented then presentation of this service component shall be started. The language of the newly selected service component shall be returned by calls from the application to the method `getCurrentLanguage` on `AudioLanguageControl` or

`SubtitlingLanguageControl` respectively. Attempting to select multiple service components of the same media type to be presented at the same time using methods on `MediaSelectControl` shall fail unless the MHP terminal concerned has sufficient resources to present them simultaneously. The failure mode shall be that as specified in `MediaSelectControl` for when insufficient resources are available.

Where methods on `MediaSelectControl` are used to modify the set of selected service components and this causes changes in the state of subtitles, the events `org.dvb.media.SubtitleSelectedEvent` and `SubtitleNotAvailableEvent` will be generated as appropriate

Control of the availability of subtitles using `MediaSelectControl` or the methods `selectLanguage` & `selectDefaultLanguage` (which `SubtitlingLanguageControl` inherits from `LanguageControl`) shall not impact the value of "Application Control" as mentioned in figure 39, "Determining subtitling language and presentation setting" on page 349 and vice versa.

## 11.4.2.8 Intersection between Streamed Media API and TV User Interface API

### 11.4.2.8.1 Basic Principles

Some of the decoder format controls pre-defined as part of `org.dvb.media.VideoFormatControl` may require control over the pixel aspect ratio of the final video output signal after video, graphics and backgrounds have been combined as shown in figure 22 on page 334. In order to enable this, a JMF player shall attempt to reserve the `HVideoDevice` instance on which its output is being displayed. Failure to reserve the `HVideoDevice` shall not be considered fatal to the JMF player but may result in an inferior TV video presentation.

NOTE: The actual configurations of the `HVideoDevice` used and available are dependent on the precise configuration of the MHP terminal. Differences will exist between set-top boxes, integrated digital TVs and PCs. The precise nature of any "inferior TV video presentation" is dependent on the configuration of the MHP terminal, on the input video and the characteristics of the final output device.

### 11.4.2.8.2 TV Behaviour Control

The "TV Behaviour Control" in figure 36, "Format control in the presence of a JMF player" on page 347, is the default video transformation behaviour for JMF players which have none other set. Setting the `VideoTransformation` to `DFC_PLATFORM` shall set the video format control to TV behaviour control mode.

NOTE 1: In this mode, any changes in the video input signal where "television behaviour control" which require changing the pixel aspect ratio of the final output signal (i.e. after the combination of video with graphics, subtitles & backgrounds) will also change the pixel aspect ratio of any graphics, subtitles & backgrounds combined into that one final output signal.

NOTE 2: For an example of "TV behaviour control" in set-top boxes and integrated digital TVs, see [E-Book \[I\]](#).

### 11.4.2.8.3 Application Behaviour Control

When an MHP application is controlling video presentation through `BackgroundVideoPresentationControl` (for video in the background) or `VideoPresentationControl` & methods on `java.awt.Component` (for video in a component), the application is responsible for monitoring the incoming video format & changing the video presentation if desired.

Instances of `VideoTransformation` constructed using the constructor of that class do not shall not impact the configuration of the `HVideoDevice`. Instances of `VideoTransformation` returned by `VideoFormatControl.getVideoTransformation` shall have the same impact on the `HVideoDevice` as the equivalent conversion applied in "TV behaviour control mode". These may not be fully achievable if the JMF player does not have the `HVideoDevice` reserved.

#### 11.4.2.8.4 Dynamic Behaviour

A JMF player shall attempt to reserve the `HVideoDevice` instance on entry to the `Prefetched` state from any state except the `Started` state. A JMF player in the `prefetched` or `started` states which does not have the `HVideoDevice` reserved shall attempt to reserve it only when the application which created the JMF player calls the `setVideoTransformation` method of the control `org.dvb.media.BackgroundVideoPresentationControl`. A JMF player leaving the `Started` or `Prefetched` state for the `Realized` state shall release the `HVideoDevice` if it has it reserved.

A JMF player which has the `HVideoDevice` reserved and then loses the right to control that device shall post an instance of the event `org.davic.media.ResourceWithdrawnEvent` to all registered listeners for `ControllerEvents` on that `Player` instance. If the right to control that device is subsequently recovered when a attempt to reserve the device (as defined above) succeeds, then an `org.davic.media.ResourceReturnedEvent` shall be posted.

#### 11.4.2.8.5 Resource Management Details

Inter-operable applications shall not call any of the methods on the instance of `ResourceClient` returned by `HVideoDevice.getClient` when a JMF player has reserved that `HVideoDevice` instance.

Inter-operable applications are allowed to call `HVideoDevice.release` or `HVideoDevice.setConfiguration` directly. JMF implementations shall tolerate this behaviour when it happens.

## 11.5 Data Access APIs

### 11.5.1 Broadcast Transport Protocol Access API

The broadcast transport protocol access API is defined in section P, "(normative): Broadcast Transport Protocol Access" on page 624.

Relative file names used to access objects in the carousel shall be taken as being relative to the base directory indicated in 10.9.2, "DVB-J application location descriptor" on page 230. Calling `new DSMCCObject(".")` or `new java.io.File(".")` will instantiate the directory object that refers to the base directory as indicated in 10.9.2, "DVB-J application location descriptor" on page 230.

### 11.5.1.1 Constraints on the java.io.File methods for broadcast carousels

The application shall be able to use the standard java.io.File class for access to broadcast carousels (e.g. a carousel unaware application). In this case, the following definitions shall apply:

- the constructor of File only creates an instance of the abstract pathname and shall not cause synchronous access to the broadcast carousel that would block the thread
- after the constructor of the DVB-J File object has been run, the directory entry information relating to this object may be in an unloaded state. The information relating to this object (e.g. its length) comes from the parent directory which is not required to have been loaded at this point by the constructor. However, this information may be available if the implementation has the information in cache.
- if the directory entry information for a DVB-J File object is in the unloaded state, then this information shall be synchronously loaded when any of the following methods are called on the object: `canRead()`, `exists()`, `isDirectory()`, `isFile()`. If the loading fails, all these methods shall return false. The effect of the loading is equivalent to calling the `synchronousLoad` method on a File object corresponding to the parent directory.
- after the constructor has been run, the content of the object may be in unloaded state. However, it may also be loaded if the implementation has the information in cache.
- if the content is in unloaded state, it shall be synchronously loaded when the `list()` method is called for a directory. If this implicit load should result in a service transfer, it shall not be done implicitly and the `list()` shall return an empty list.
- the method `lastModified()` returns `moduleVersion` from the DII for the module that carries the file (treating the octet as an unsigned integer).
- any version changes in a file after the constructor for an `InputStream` is called will not be visible in the data read from that `InputStream`.
- list methods shall return null when called on a object that isn't a directory.
- the creation of a `FileInputStream` shall throw `FileNotFoundException` if the referenced object is not a file.
- Failure of a signed file to be authenticated shall be reported by the constructor for the `InputStream` throwing an `IOException`.
- Failure of a signed directory to be authenticated shall be reported by the list methods returning null when called on a file object representing the directory whose authentication failed.
- The value returned by `java.io.file.Length()` may not be accurate as it returns the value from the directory information rather than the actual size of the file

There are no guarantees that the most recent version of a file will be returned unless the network signalling specifies that the file concerned requires transparent access.

### 11.5.1.2 Methods dealing with write access

The `java.io.File` class also contains methods that assume write access to the file system. Due to its broadcast nature, the receiver naturally does not have write access to the carousel. It should be noted, however, that a broadcast carousel is not a read-only file system (which has the property of not changing). The carousel content can certainly be written and modified, but only by broadcaster - not the receiver. Therefore, the situation is equivalent to a Unix file system where the user has only read permissions, but not write permissions or ownership of the files.

The following `java.io.File` methods deal with write access to directories: `canWrite()`, `mkdir()`, `makedirs()`, `renameTo()`.

For abstract pathname entries in the broadcast carousel, the following behaviour shall apply:

- `canWrite()` returns false to indicate that the file can not be written to
- `mkdir()`, `makedirs()` and `renameTo()` return false to indicate that the request failed

### 11.5.1.3 Behaviour following loss of a broadcast file system

When a broadcast file system is lost to an application as described in 6.2.5.3, "Loss of Carousel Behaviour" on page 56, the following failure modes shall be followed for data which is unavailable.

- Attempting to load data from a file using a content format specific API shall fail as if the file itself never existed. This shall be done according to the specific API concerned.
- The classloader created by the platform when the application was first launched never attempts to automatically recover following loss of its initial broadcast file system.
- Attempting to load a class which is unavailable shall fail as if the class was never present.
- After the constructor for an `InputStream` has been successfully called, the data for that `InputStream` shall be maintained by the platform until the `InputStream` is closed. This is the same behaviour as specified for `InputStream` and file version changes in 11.5.1.1, "Constraints on the `java.io.File` methods for broadcast carousels" on page 256.
- Attempting to create a `FileInputStream` for a file whose data is unavailable shall fail with a `FileNotFoundException`.
- Attempting to create a `RandomAccessFile` for a file whose data is unavailable shall fail with an `IOException`.
- Attempting to use a `FileDescriptor` of a `FileInputStream` whose data is unavailable shall result in a `FileInputStream` or `InputStreamReader` where all methods shall throw an `IOException`.
- Attempting to call methods on a `File` object whose data is unavailable shall fail in the same way as if the `File` itself never existed.

### 11.5.2 Support for Multicast IP over the Broadcast Channel

Where support for IP over the broadcast channel is included, the following classes and packages shall be supported in addition to those listed above for the case where IP support is not included.

- The `javax.tv.net` package as defined in Java TV [51] is included.
- The methods listed below from `java.net.MulticastSocket` shall be supported for one way IP Multicast support. All other methods in the class shall either throw exceptions where possible or fail silently when called on receive only IP Multicast systems.

```
MulticastSocket()
MulticastSocket(int port)
InetAddress getInterface()
void joinGroup(InetAddress mcastaddr)
void leaveGroup(InetAddress mcastaddr)
void setInterface(InetAddress inf)
```

- The methods listed below from `java.net.DatagramSocket` shall be supported for one way IP Multicast support. These methods are inherited by `MulticastSocket`. All other methods in the class shall throw exceptions where possible or fail silently when only called on receive only IP Multicast systems.

```
DatagramSocket()
DatagramSocket(int port)
DatagramSocket(int port, InetAddress local)

void close()
int getSoTimeout()
void receive(DatagramPacket p)
void setSoTimeout(int timeout)
int getLocalPort()
InetAddress getLocalAddress()
```

d) The following classes from `java.net` shall be supported:-

- `DatagramPacket`
- `SocketException`
- `UnknownHostException`
- `ProtocolException`
- `BindException`

e) The class `org.dvb.net.DatagramSocketBufferControl` shall be supported see Q, "(normative): [Datagram Socket Buffer Control](#)" on page 685.

f) Behaviour if unsupported:

When the application tries to `joinGroup()` on a Multicast address, `ProtocolException` shall be thrown to indicate failure on joining the group.

### 11.5.3 Support for IP over the Return Channel

Where support for IP over the return channel is included, all of the `java.net` package shall be included. Platforms not implementing specific optional return channel protocols shall fail as defined in the specification of this API.

On devices whose return channel can be connected or disconnected, connecting a `java.net.Socket` or a `java.net.URLConnection` to a host addressed via the return channel shall automatically setup a connection to the default connection target subject to the application having return channel permission for the default ISP. Such connections shall be automatically disconnected after a time out period defined in the Navigator.

See 11.10, "[Java permissions](#)" on page 272.

See also 11.8.1.3, "[Other classes](#)" on page 269.

### 11.5.4 MPEG-2 Section Filter API

The MPEG-2 section filter API is defined in annex E of [DAVIC 1.4.1p9 \[3\]](#).

### 11.5.5 Mid-Level Communications API

See Annex R, "(normative): [DVB-J Return Channel Connection Management API](#)" on page 688.

### 11.5.6 Persistent Storage API

The API to persistent storage shall be the `java.io` package and the extensions to it found in the `org.dvb.io.persistent` defined in annex J, "(normative): [DVB-J persistent storage API](#)" on page 469.

The "`dvb.persistent.root`" property which can be obtained from `java.lang.System.getProperty()` identifies the directory at the root of the file name space used for persistent storage. This value shall be the same for all applications. Accessing any files or directories in the parent directory of this root or directories above that shall throw a `SecurityException` as defined in the specification for the `java.io` package. Applications shall have read only access to this root directory. Applications shall have automatic read and write access to the sub-directories `<organisation_id>` and `<organisation_id> + <file.seperator> + <application_id>`. See 14.5, "[Text encoding of application identifiers](#)" on page 357. The fields `<organisation_id>` and `<application-id>` are hexadecimal text encodings (without leading zeros) of the fields in the 48 bit application identifier of the application concerned as defined in 10.5, "[Application identification](#)" on page 214. These directories must be automatically created by the platform. The owner of the organisation directory shall be the platform and it shall always have organisation read / write and world read access. The platform shall set the owner of the application directory to the application with read and write access for that application and no access for other applications.

All files in persistent storage shall store the 48 bit application identifier of their creator as the "owner" of the file. Only the owner of the file is entitled to change the file attributes and file access rights. The owner of a file cannot be changed once a file is created. Applications may only create files or directories in directories to which they have write access. The existing semantics for the `java.io` package are respected.

## 11.6 Service Information and Selection APIs

### 11.6.1 DVB Service Information API

The DVB specific SI API is defined in annex M, "(normative): SI Access API" on page 490.

### 11.6.2 Service Selection API

The service selection API is defined by the `javax.tv.service.selection` package from Java TV [51].

On the first occasion when the method `ServiceContext.getServiceContentHandlers` is called for a specific service, any JMF players returned shall always be in the started state and if they are presenting video, that video shall always be presenting on the background video device.

Applications shall re-acquire the list of service content handlers after a service selection completes. It is implementation dependent whether any previous handlers are re-used. JMF players which are not re-used are stopped and may be disposed by the platform. Where a JMF player is re-used, it shall be reset to a default condition as if it was not being re-used. In particular the implementation shall cancel all listeners previously defined and reset any previously defined characteristics such as video scaling or positioning.

The exception `javax.tv.services.selection.ServiceContextException` shall never be thrown by MHP terminals.

The default media player behaviour following service selection is defined under 11.4.2.3, "Default media player behaviour" on page 251.

The protected constructor of `ServiceContextFactory` is for implementation use. MHP applications shall not subclass `ServiceContextFactory`. Implementations are not required to behave correctly if they should do this.

Implementations of this API are required to perform tuning as part of implementing the `ServiceContext.select()` method where the new service to be selected is part of a transport stream known to the MHP terminal but not currently tuned to.

This API shall support the following features on those MHP terminals which support the feature concerned.

- Internet clients (see: 9.8, "Lifecycle of internet access applications" on page 206, 11.14.1, "Internet client control APIs" on page 283)
- Stored services (see: 9.7.2, "Stored services" on page 204, 11.12.2.2, "Modified behaviour of MHP 1.0 APIs" on page 281)
- Services signalled over the interaction channel (see 9.7.1, "Applications loaded from the interaction channel" on page 204, 11.11.12, "Support for the HTTP protocol in DVB-J" on page 280)

### 11.6.3 Tuning API

The tuning API is defined in annex H of DAVIC 1.4.1p9 [3] apart from section H.4, (the `Locator` and `DvbLocator` classes) which are found in section 11.7.6 "Content Referencing".

The following methods in this package shall throw `java.lang.SecurityException` where and only where the application does not have a `org.dvb.net.tuning.TunerPermission`:

- `NetworkInterfaceController.reserve`
- `NetworkInterfaceController.reserveFor`

The class `org.davic.net.tuning.dvb.DvbNetworkInterfaceSIUtil` is not required.



See also 11.8.3, "Additional permissions classes" on page 269.

## 11.6.4 Conditional Access API

The conditional access API is defined in annex I of [DAVIC 1.4.1p9 \[3\]](#).

The following classes are not supported as defined in section 11.2.2 "Approach to Subsetting" - CA1Module, CA0Module, CA1Message, CA0Message, CA1ModuleResponseEvent and CA0ModuleResponseEvent.

Physical CI modules or embedded systems following the CI protocol can produce MMI messages. The API implementation, subject to security model, passes those to be presented by the application if the application is interested. Otherwise CA dialogs are generated.

The following methods in this API may throw `java.lang.SecurityException`:-

- `CAModuleManager.addMMIListener`
- `CAModule.queryEntitlement`
- `CAModule.listEntitlements`
- `CAModule.buyEntitlement`
- `CAModule.openMessageSession`

The `sessionIDs` used in CI message passing are unique to a single application and access shall fail if shared between applications.

See also 11.8.3, "Additional permissions classes" on page 269.

## 11.6.5 Protocol Independent SI API

The protocol independent SI API is defined by the following packages from [Java TV \[51\]](#):

- `javax.tv.service`
- `javax.tv.service.guide`
- `javax.tv.service.navigation`
- `javax.tv.service.transport`

The mapping of this onto the DVB-SI protocol is specified in annex O, "(normative): [Integration of the JavaTV SI API and DVB SI](#)" on page 617.

Cancellation shall fail if the request is no longer pending. It is implementation dependent if cancellation fails under any other circumstances. Implementations are not required to support cancellation of requests between when the requested SI data has arrived in the device and when execution of the `notifySuccess` method starts.

The interface `javax.tv.service.ServiceMinorNumber` is not required to be implemented by any object defined in this specification.

This API shall support the following features on those MHP terminals which support the feature concerned.

- Internet clients (see 9.8, "Lifecycle of internet access applications" on page 206, 11.14.1, "Internet client control APIs" on page 283)
- Stored services (see 9.7.2, "Stored services" on page 204, 11.12.2.2, "Modified behaviour of MHP 1.0 APIs" on page 281)



## 11.7 Common Infrastructure APIs

### 11.7.1 APIs to support DVB-J application lifecycle

This API is formed of the Java classes and interfaces found in the `javax.tv.xlet` package specified in [Java TV \[51\]](#).

#### 11.7.1.1 Xlet properties

The following named Xlet properties are introduced:

- `dvb.org.id`
- `dvb.app.id`
- `dvb.caller.parameters`

They can be retrieved from an Xlet's `XletContext` by calling `getXletProperty` with the string name defined above.

All keys for `XletContext.getXletProperty` beginning 'dvb.' are reserved for use in DVB project specifications.

The array of strings returned by `XletContext.getXletProperty(XletContext.ARGS)` shall be the array of strings defined in the [DVB-J application descriptor](#) (see [10.9.1 on page 230](#)) in the same order as specified in the signalling. Each entry in the loop of that descriptor shall be presented as one string in the array returned by this method, interpreted using the encoding as specified in [10.4.8, "Text encoding in AIT" on page 213](#). Zero-length strings in the signalling shall be represented as the empty string.

#### 11.7.1.2 Actions for DVB-J applications to perform in their destroy method

Xlets shall perform at least the following in their `Xlet.destroyXlet` method and before calling `XletContext.notifyDestroyed`:

- cause any threads that they have created to exit voluntarily. See ["java.lang package" on page 243](#).
- stop, deallocate and close any JMF players that they have created.
- stop and destroy any JavaTV service selection `ServiceContext` objects that they created.
- release any other scarce resources that they created, e.g. `NetworkInterfaceControllers` if they do any tuning.
- flush any images using the `Image.flush()` method.
- Xlets shall not cause any unnecessary delay in their `Xlet.destroyXlet` method.
- de-register any event listeners

### 11.7.2 Application discovery and launching APIs

This API is formed of the `org.dvb.application` package defined in [S, "\(normative\): Application Listing and Launching" on page 711](#).

The following properties are defined for use with the method `AppAttributes.getProperty`:

**Table 114 : Application attribute properties (Sheet 1 of 2)**

Property name (note 1)	Return
<code>dvb.j.location.base</code>	Returns String containing <code>base_directory_bytes</code> from DVB-J application location descriptor.
<code>dvb.j.location.cpath.extension</code>	Returns <code>String[]</code> derived from <code>classpath_extension_bytes</code> of DVB-J application location descriptor with each array entry corresponding to a pathname entry as defined for <code>classpath_extension_bytes</code> .

**Table 114 : Application attribute properties (Sheet 2 of 2)**

Property name (note 1)	Return
dvb.transport.oc.component.tag	Returns Integer containing the <a href="#">component_tag</a> from the selector bytes of the transport protocol descriptor
NOTE 1: Property names beginning "dvb." are reserved for future use.	

The following table defines the source of the information which shall be used for methods returning information from entries in the application database for an application signalled in an AIT. This shall apply to both AITs delivered via MPEG-2 broadcast and ones delivered as an AIT file (see 10.4.9, "AIT file" on page 213). This shall also apply to applications running as part of a stored service using information stored at the time when the application itself was stored.

**Table 115 : Information source for methods on AppAttributes**

Method	Information source
getName()	One of the names that can be found in the application name descriptor.
getName(String ISO639code)	A name of the application from the application name descriptor corresponding to the specified language.
getNames()	All of the names for the application which can be found in the application name descriptor and their ISO 639 language code.
getProfiles()	The set of profiles signalled in the application profile field of the application descriptor.
getPriority()	The contents of the application_priority field of the application descriptor.
getVersions(String profile)	The values version.major, version.minor and version.micro for the specified profile from the application descriptor.
getIsServiceBound()	True if the service_bound field in the application descriptor is set to 1. Otherwise false.
isStartable()	An application shall be considered to be transmitted on a remote connection where the application control code in the signalling is REMOTE. An application shall be considered to be not available any more where the corresponding entry has been removed from the AIT and the implementation has detected this.
getIdentifier()	The contents of the application_identifier field of the application information section.
getServiceLocator()	An application shall be considered to be transmitted on a remote connection only where the application control code in the signalling is REMOTE. The locator for a remote application shall encapsulate the values found in the selector bytes of the transport protocol descriptor.

**Table 116 : Information source for methods on Applcon**

Method	Information source
getLocator()	The bytes carried in the application_locator_byte of the application icons descriptor appended to either the base directory of the application (where the application type from the application information section is zero, from the DVB-J application location descriptor) or the physical root (where the application type from the application information section is 1, from the DVB-HTML application location descriptor).
getIconFlags()	The icon_flags field of the application_icon_descriptor.

Applications signalled in an AIT shall be considered to be externally authorised where their only signalling in that AIT is by an [External application authorisation descriptor](#).

`CurrentServiceFilter` shall behave as defined in its specification for stored services and services signalled over the interaction channel in addition to broadcast service as defined in its specification.

### 11.7.3 Inter-Application and Inter-Xlet communication API

The javadoc for the `org.dvb.io.ixc` package is provided in annex Y, "(normative): Inter-application and Inter-Xlet communication API" on page 1261. An example is provided in W.2, "Example of exporting an object for inter-application communication" on page 1251.

This API is formed of the interfaces `java.rmi.Remote` and `java.rmi.RemoteException`, and the classes `java.rmi.NotBoundException` and `java.rmi.AlreadyBoundException` as specified in JAE 1.1.8 API [31], plus the package `org.dvb.io.ixc`.

Two named Xlet properties are introduced: `dvb.org.id` and `dvb.app.id`. They can be retrieved from an Xlet's `XletContext` by calling `getXletProperty("dvb.org.id")` and `getXletProperty("dvb.app.id")`, respectively.

#### 11.7.3.1 Remote Call Semantics

An object may be communicated to another Xlet in two ways:

- A reference to the remote object can be passed,
- or
- a copy of the remote object can be made.

These two techniques are called "pass by remote reference", and "pass by remote copy". When an object that has been bound to the IXC Registry via a method of `org.dvb.io.ixc.IxcRegistry` is imported by another, it shall be passed by remote reference.

##### 11.7.3.1.1 Objects Passed by Remote Reference

An object that is passed by remote reference must implement a remote interface. A remote interface is an interface that implements the marker interface `java.rmi.Remote`. The declared type of a parameter or a return value for a remote method invocation must be a remote interface, or a class whose instances are serializable. If a remote interface that is application-defined, the interface definition must be included in both the sending and receiving Xlet. If the two xlets contain an identically named remote interfaces that contain different declarations, the result of attempting to use these interfaces for inter-Xlet communication is undefined, and possibly implementation dependent.

When an object is passed by remote reference to a different Xlet, the receiving Xlet does not receive a direct reference to the exported object; rather, it receives an instance of a stub class. This stub class will not be a subclass of the remote object's runtime type; rather, it will be a platform-generated class that implements the remote interface type that is the declared type of the argument or return value. It will include implementations of all methods declared in the remote interface, and will contain no other members accessible to the application. Remote methods invoked on this stub class instance will be forwarded to the object in the original Xlet, and executed in the context of that Xlet.

The definition of the stub class shall be automatically created by the platform.

NOTE: This differs from traditional network RMI, where the stub classes are created by the developer using a tool such as `rmic`. If stub classes produced by `rmic` or any other off-line tool are present, the platform shall silently ignore them for the purposes of inter-Xlet communication.

The stub class that is generated shall include a definition for all of the remote methods defined on the class of the original object. A remote method is a method declared in a remote interface that is implemented by the given class. A remote interface is an interface that extends `java.rmi.Remote`, either directly or indirectly. Methods declared in a remote interface must be defined as follows:

- Each method must declare `java.rmi.RemoteException` in its `throws` clause, in addition to any application-specific exceptions.
- A remote object passed by remote reference as an argument or return value must be declared as an interface that extends `java.rmi.Remote`, and not as an application class that implements this remote interface.
- The type of each method argument must either be a remote interface, a class or interface that implements `java.io.Serializable`, or a primitive type.
- Each return value must either be a remote interface, a class or interface that implements `java.io.Serializable`, a primitive type, or `void`.

If any remote method does not follow these rules, the platform cannot generate a stub class. When one is required, a `RemoteException` shall instead be thrown to the caller.

#### 11.7.3.1.1.1 Lifecycle Considerations for Remote Objects

When an Xlet is destroyed, it is possible that other Xlets may have remote references to some of the Xlet's objects. If a method is invoked on one of these remote objects, the platform may fail to execute the method, and instead throw a `RemoteException`. If a remote method call is in progress when the Xlet receiving the call is destroyed, the calling Xlet may receive a `RemoteException` on the calling thread, and the remote method invocation may be abruptly terminated. If a remote method has started executing code in the implementation of the remote object when the Xlet making the call is destroyed, the call shall run to completion, unless the Xlet receiving the call is also destroyed.

#### 11.7.3.1.1.2 Exceptions in Remote Method Calls

If an exception is thrown from a remote method, a remote copy of that exception shall be made in the context of the calling Xlet. This copy of the exception shall be thrown to the caller.

#### 11.7.3.1.1.3 Re-exported Objects

It is possible that an object passed from one Xlet to another might be passed back to the original Xlet. This could happen through any number of intervening Xlets. If this happens, the original Xlet will receive the instance that it originally exported. If it compares the instance it receives with the original instance using the Java `==` operator, the result will be true. Because of this, there is no need to provide an override of `java.lang.Object.equals()` or `java.lang.Object.hashCode()` for remote objects.

**NOTE:** This behaviour is different than network RMI, as implemented in traditional Java implementations. In Sun's implementation of network RMI, a remote stub object is given to the original Xlet, but stubs and remote objects are required to have a special version of the `equals()` and `hashCode()` methods.

#### 11.7.3.1.2 Objects Passed by Remote Copy

An object is passed by remote copy when a method argument or return value is passed, where the class of that object does not implement `java.rmi.Remote`. Additionally, a remote method call exception is communicated to the receiving Xlet by remote copy, as described in 11.7.3.1.1.2, "Exceptions in Remote Method Calls" on page 264.

When an object is passed by remote copy, it is serialized into a byte stream in the context of the exporting Xlet, and deserialized in the context of the importing Xlet. Serialization is performed as defined for `java.io.Serializable`. Application-defined classes may be serialized, but the definition of the application-defined class must be present in both Xlets, and the external forms of both versions of the class must be compatible. If any error in serialization or deserialization occurs, an instance of `java.rmi.RemoteException` shall be thrown.

#### 11.7.3.1.2.1 Treatment of Primitive Types

Primitive types passed as method arguments or return values are copied.

### 11.7.3.1.3 Classloading Considerations

The presence of inter-xlet communication does not allow the loading of one Xlet's classes from another. No classloader that loads classes from a remote Xlet for remote method calls is created (unlike network RMI, which creates a special `RMIClassLoader` for remote objects). Rather, a copy of each application-defined remote interface and serializable object involved in a remote method invocation must be present in both Xlets. If this is not the case, the platform shall generate a `RemoteException` and throw it in the calling thread.

### 11.7.3.1.4 Thread Usage

A remote method may or may not execute in separate underlying thread. If an application makes a remote method invocation to a remote object in a different application, and that second application calls back to the first in the same "thread," then the first application might or might not observe that the original calling thread and the callback thread are the same instance of `java.lang.Thread`.

If an application makes simultaneous remote calls in separate threads, then the remote execution shall appear to be carried out in parallel.

NOTE: This is not meant to rule out thread-pooling techniques. Specifically, an implementation may choose to serialize such remote calls, as long as the first one completes within a reasonably short time, relative to the normal scheduling rules of Java threads.

### 11.7.3.1.5 Garbage Collection of Remote Objects

When a non-destroyed Xlet contains a reachable instance of a stub for a remote object, that remote object shall not be garbage collected, unless the remote Xlet is destroyed. When an exported object no longer has any remote stub objects that are reachable in other non-destroyed Xlets, and when that exported object is also not reachable locally within its Xlet, then that remote object shall be considered unreachable, and thus eligible for reclamation.

When an Xlet is destroyed, other Xlets may hold remote references to objects within the Xlet being destroyed. In this case, the referenced objects may be dereferenced and ultimately garbage collected. If this is done, then attempts to invoke remote methods on these objects shall result in a `RemoteException` to the caller.

## 11.7.4 Basic MPEG Concepts

This API is formed of the Java classes defined in annex G of [DAVIC 1.4.1p9 \[3\]](#):

- `ApplicationOrigin`,
- `ElementaryStream`,
- `Service`,
- `TransportStream`,
- `DvbElementaryStream`,
- `DvbService`,
- `DvbTransportStream`.

Methods returning instances of elementary stream, service or transport stream shall return instances of the dvb specific subclass (`DvbElementaryStream` etc.).

## 11.7.5 Resource Notification

The resource notification API is defined in annex F of [DAVIC 1.4.1p9 \[3\]](#).

- The `notifyRelease()` method shall be called for all `ResourceClients` where the corresponding resource has been removed without the application releasing it. The `release()` method shall be called for specific resources where time to cleanup is of practical use considering the specific resource in question. The only one of these in this specification is connection oriented return channels.
- The `requestData` parameter of the `requestRelease` method shall implement the `java.rmi.Remote` interface. In APIs using the `ResourceProxy` interface, where the method to reserve the resource has a parameter 'requestData', this parameter shall also implement the `java.rmi.Remote` interface or be `null`. Use of other values shall result in `null` being used when `requestRelease()` is called. Implementing the `Remote` interface does not force `java.rmi` to be used when the current owner of the resource is in the same application as code requesting ownership.
- The `resourceProxy.getClient()` method shall always return the `ResourceClient` provided to the platform by the application when that instance of a `ResourceProxy` was created or reserved. The `ResourceClient` for the current owner of a resource is only returned when this method is called on the `ResourceProxy` which currently holds the resource concerned.

This text clarifies the DAVIC javadoc with respect to multiple simultaneous applications. Interpretations of the DAVIC javadoc which are inconsistent with this are excluded.

## 11.7.6 Content Referencing

This API is formed of the classes found in section H.4 of annex H of [DAVIC 1.4.1p9 \[3\]](#) - the `Locator` and `DvbLocator` classes. It also includes the `javax.tv.locator` package as defined in [Java TV \[51\]](#).

The signature of the `org.davic.net.Locator` class will be extended with:

```
"implements javax.tv.locator.Locator"
```

The `createFactory()` method of `javax.tv.locator.LocatorFactory` shall always return `org.davic.net.Locator(s)` which implement the `javax.tv.locator.Locator` interface when provided with DVB URLs as input (as defined in 14.1, "Namespace mapping (DVB Locator)" on page 353).

In this specification, methods whose signature has a return type of `org.davic.net.Locator` or `javax.tv.locator.Locator` shall return an instance of `org.davic.net.dvb.DvbLocator` (or a platform defined subclass of that) where the locator returned can be represented by the DVB locator syntax described in [DAVIC 1.4.1p9 \[3\]](#). In this case, the `DvbLocator` returned shall contain the numeric identifiers of a DVB service (see 14.9, "Service identification" on page 359).

In `javax.tv.locator.Locator.toExternalForm()`, the canonical form of a DVB locator is defined as follows:

- For instances of `org.davic.net.dvb.DvbNetworkBoundLocator`, this should be the format defined in the MHP specification, including the transport stream id.
- For instances of `org.davic.net.dvb.DvbLocator` which are not instances of the above sub-class, this should be the format defined in the MHP specification, excluding the transport stream id.

Any optional extensions (for specifying components, events etc.) are considered in a comparison and if they are not equally present in both locators then the comparison shall fail.

For the above locators "best effort" comparison shall be exact.

The protected constructor of `LocatorFactory` is for implementation use. MHP applications shall not subclass `LocatorFactory`. Implementations are not required to behave correctly if they should do this.

## 11.7.7 Common Error Reporting

This API is formed of the interface and exceptions defined in annex G of [DAVIC 1.4.1p9 \[3\]](#):

- `NotAuthorizedInterface`,
- `NotAuthorizedException`,
- `ObjectUnavailableException`,
- `ResourceException`,
- `TuningException`.

## 11.7.8 Plug-in APIs

The plug-in API is defined in Annexes AF, "(normative): Plug-in APIs" on page 1328 and AE, "(normative): Inner Applications" on page 1314 of this specification.

MHP inter-operable plug-ins signalled with the [delegated application descriptor](#) (see [10.13.3 on page 236](#)) shall implement the `org.dvb.application.plugins.Plugin` interface. Failure to implement this interface shall result in the plug-in application being ignored. Plug-in applications which only execute "delegated" applications signalled in the AIT do not need to implement the `javax.tv.xlet.Xlet` interface as part of the initial entry point from the application manager to the plug-in. Plug-in applications which can execute both AIT & natively signalled "delegated" applications shall implement both the Plug-in and the Xlet interfaces.

The semantics of the method `org.dvb.application.AppAttributes.getProperty()` are extended such that when it is called with the string "dvb.ait.descriptors", an array of bytes containing the `application_descriptors_loop` shall be returned. If this information is no longer available then null shall be returned.

Each running instance of a plug-in application shall execute in a single logical VM instance as defined in [11.2.1, "Basic Considerations" on page 241](#). Hence all delegated application Xlets returned by the `initApplication` method are all executed in the same logical VM instance. It is the responsibility of the plug-in application to ensure the presence or absence of shared context between instances of delegated application Xlets in the same logical VM instance.

## 11.8 Security

### 11.8.1 Basic Security

The following packages and classes as defined in [PersonalJAE \[36\]](#) are supported.

#### 11.8.1.1 java.security

From the `java.security` package:

- `AccessControlContext`
- `AccessControlException`
- `AccessController`
- `AllPermission`
- `BasicPermission`
- `CodeSource`
- `GeneralSecurityException`
- `Guard`
- `InvalidKeyException`
- `Key`
- `KeyException`
- `NoSuchAlgorithmException`
- `NoSuchProviderException`
- `Permission`
- `PermissionCollection`
- `Permissions`
- `Policy`
- `Principal`
- `PrivilegedAction`
- `PrivilegedActionException`
- `PrivilegedExceptionAction`
- `ProtectionDomain`
- `PublicKey`
- `SignatureException`
- `SecurityPermission`
- `SecureClassLoader` (note 1)
- `UnresolvedPermission`

NOTE 1: The `java.security.SecureClassLoader` class extends the JDK 1.1.8 version of `java.lang.ClassLoader`, therefore the `java.security.SecureClassLoader` class is not required to support additional APIs from later versions of `java.lang.ClassLoader`.

The MHP platform shall always install a `SecurityManager` before starting any DVB-J applications.

The following classes shall be supported:

- `java.security.KeyFactory`

`KeyFactory.getInstance(String algorithm)` shall return a valid `KeyFactory` for algorithm "RSA".

NOTE: since `KeyFactorySpi` is not required, the protected constructor will be subset out via the subsetting rule

- `java.security.spec.KeySpec`
- `java.security.spec.EncodedKeySpec`

The `EncodedKeySpec` constructor is guaranteed to work for X.509 key encodings.

- `java.security.spec.X509EncodedKeySpec`



### 11.8.1.2 `java.security.cert`

- `Certificate`
- `CertificateEncodingException`
- `CertificateException`
- `X509Certificate`
- `CertificateExpiredException`
- `CertificateNotYetValidException`

### 11.8.1.3 Other classes

The following other classes are supported.

- `java.io.FilePermission`
- `java.io.SerializablePermission`
- `java.lang.RuntimePermission`
- `java.util.PropertyPermission`
- `java.net.SocketPermission`
- `java.awt.AWTPermission`

## 11.8.2 APIs to Support TLS / SSL Over the Return Channel

This API is defined in the following packages from JSSE [60]:

- `javax.net`
- `javax.net.ssl`
- `javax.security.cert`

Also see 12.10, "Security on the return channel" on page 320.

### 11.8.3 Additional permissions classes

See T, "(normative): Permissions" on page 746.

### 11.8.4 General security issues

Unless otherwise specified, sub-classes of `java.security.Permission` are not required to check the input parameters to their constructors. When an MHP platform constructs these, it is responsible for creating them with legal values. The behaviour of an MHP platform if an application creates such an instance with illegal values is intentionally not specified.

## 11.9 Other APIs

### 11.9.1 Timer Support

This API is formed of the Timer API defined in Java TV [51] in the `javax.tv.util` package.

Implementations are required to meet the following specifications:

- Minimum repeat interval less than or equal to 40 ms
- Granularity less than or equal to 10ms

The only condition defined in this specification where `TVTimer.scheduleTimerSpec` may throw `TVTimerScheduleFailedException` is if the MHP terminal is unable to provide any more timers. See table G.4, "Minimum requirements for other resources" on page 447.

## 11.9.2 User Settings and Preferences API

This API is defined in L "(normative): User Settings and Preferences API".

The preferences listed below shall be accessible to unauthorised applications.

- User Language
- Parental Rating
- Country Code
- Default Font Size

Other preferences shall not be accessible to unauthorised applications.

## 11.9.3 Profile and version properties

Applications can discover the supported profile and the version of the profile (and thus, what functionality is supported) by retrieving profile and version properties. If a particular profile is not supported then the related version properties shall return null.

More specifically, the properties listed in table 117 shall be included in the property set of the `java.lang.System` class. Thus these properties can be retrieved using `java.lang.System.getProperty()`. Since this API returns a string, numeric return values shall be encoded as defined by `java.lang.Integer.toString(int)`.

**Table 117 : System properties for profile and version interrogation (Sheet 1 of 2)**

Property	Semantics	Possible values	Example
<code>mhp.profile.enhanced_broadcast</code>	Indicates whether the enhanced broadcast profile is supported	"YES", "NO"	"YES"
<code>mhp.profile.interactive_broadcast</code>	Indicates whether the interactive broadcast profile is supported	"YES", "NO"	"NO"
<code>mhp.profile.internet_access</code>	Indicates whether the internet access profile is supported	"YES", "NO"	"NO"
<code>mhp.eb.version.major</code>	Major version number of the supported enhanced broadcast profile	Non-negative integer value	"1"
<code>mhp.eb.version.minor</code>	Minor version number of the supported enhanced broadcast profile	Non-negative integer value	"0"
<code>mhp.eb.version.micro</code>	Micro version number of the supported enhanced broadcast profile	Non-negative integer value	"0"
<code>mhp.ib.version.major</code>	Major version number of the supported interactive broadcast profile	Non-negative integer value	"1"
<code>mhp.ib.version.minor</code>	Minor version number of the supported interactive broadcast profile	Non-negative integer value	"0"
<code>mhp.ib.version.micro</code>	Micro version number of the supported interactive broadcast profile	Non-negative integer value	"0"
<code>mhp.ia.version.major</code>	Major version number of the supported internet access profile	Non-negative integer value	"1"

**Table 117 : System properties for profile and version interrogation (Sheet 2 of 2)**

Property	Semantics	Possible values	Example
mhp.ia.version.minor	Minor version number of the supported internet access profile	Non-negative integer value	"0"
mhp.ia.version.micro	Micro version number of the supported internet access profile	Non-negative integer value	"0"

The properties for querying the version of a profile which is not supported by the MHP terminal shall not be supported. Hence, calling `System.getProperty()` for these properties shall return null.

### 11.9.3.1 Information on options

Chapter 15 defines what is mandatory and optional for each profile. In order to give an application information about which options a particular MHP implementation supports, a property string is defined for each option (with the same granularity as in chapter 15).

An MHP implementation supports an option if and only if the corresponding property is known and its value is "SUPPORTED". The properties are part of the property set of the `java.lang.System` class.

The general syntax of the properties that indicate whether a certain feature is supported or not is:

```
mhp.option.<the optional feature>.
```

The table 118 lists the currently defined options.

**Table 118 : System properties for optional feature interrogation**

Option	Property
IP Multicast over Broadcast Channel	mhp.option.ip.multicast
HTTP 1.1 over the interaction channel	mhp.option.http.1.1
DSMCC user-to-user over the interaction channel	mhp.option.dsmcc.uu
DVB-HTML	mhp.option.dvb.html

## 11.9.4 Non-CA smart card API

### 11.9.4.1 Framework

The non conditional access API for smart cards is defined in [Embedded OCF \[92\]](#) with the clarifications, extensions and restrictions as defined in the corresponding sections below.

### 11.9.4.2 Clarifications

The implementation has to make sure that there is at most one open `SlotChannel` per physical slot channel at any given time.

The use of `opencard.opt.terminal.Lockable` is not applicable in the MHP context.

### 11.9.4.3 Extensions

This API is extended with the classes found in Annex AI, "(normative): Non-CA Smartcard APIs" on page 1396.

### 11.9.4.4 Restrictions

The behaviour is changed so that an application only can obtain instances of card services that are either registered by the system or that it itself has registered.

`CardServiceScheduler.getSlotChannel()` shall return null, this is to prevent application developers to be tempted to bypass the use of a `CardChannel` in services.

The `opencard.opt.terminal` package is not required in MHP.

### 11.9.4.5 Modified semantics

The following methods shall throw `java.lang.SecurityException` if the calling application does not have the `SmartCardPermission` class defined in 11.9.4.3, "Extensions" on page 271.

- `CardTerminal.getTerminal`
- `EventGenerator.getGenerator`
- `SmartCard.getSmartCard`
- `CardServiceRegistry.getRegistry`

If no non-CA smart card reader is available to MHP applications, the following failure behaviour shall be observed.

- `CardTerminal.getTerminal()` shall return a `CardTerminal` instance where the `getSlots` method shall return zero. The other methods shall fail according to their specification where the `slotID` is invalid.
- The methods on `EventGenerator` shall work but the MHP terminal shall never generate any events to any registered listeners.
- `SmartCard.getSmartCard` shall throw a `CardTerminalException`
- `CardServiceRegistry.getRegistry` shall return a `CardServiceRegistry` which does not initially have any `CardServiceFactories` in it. Applications are able to add factories by the `add` method but the platform is not required to use them for any purpose other than returning them in the enumeration from `getCardServiceFactories`.

## 11.10 Java permissions

This section explains how the permissions that are defined to be included in the sandbox that is available to unsigned applications and the permissions that can be requested in the permission request file are mapped to the `Permission` objects in the Java platform.

### 11.10.1 Permissions for unsigned applications

The MHP security policy includes a set of resources that are always guaranteed to be granted to applications, if the application is executed. Unsigned applications have access to only these resources.

This section defines the mapping of those resources to the Java `Permission` objects. DVB-J applications shall always be granted these `Permissions`.

#### 11.10.1.1 `java.awt.AWTPermission`

This control access to sensitive parts of AWT which is not needed for MHP applications. This shall be denied for both unsigned and signed applications.

#### 11.10.1.2 `java.net.SocketPermission`:

Because access to return channel is not within the sandbox, this is not required for unsigned applications.

#### 11.10.1.3 `java.util.PropertyPermission`

For unsigned applications, a read permission shall be granted for all properties defined in this chapter except for those explicitly identified as only available for signed applications. The permission shall be denied for the action string "write".

#### 11.10.1.4 `java.lang.RuntimePermission`

This permission shall be denied for both unsigned and signed applications.

#### 11.10.1.5 `java.io.SerializablePermission`

This permission shall be denied for both unsigned and signed applications.

### 11.10.1.6 `java.io.FilePermission`

A read permission shall be granted for the subtree under which the implementation mounts the object carousels.

### 11.10.1.7 `javax.tv.media.MediaSelectPermission`

The Media API (i.e. JMF) is within the sandbox, so all applications shall be granted a `javax.tv.media.MediaSelectPermission` with a locator string "\*" that indicates access to all media streams.

### 11.10.1.8 `javax.tv.service.ReadPermission`

Access to Service Information is within the sandbox, so all applications shall be granted a `javax.tv.service.ReadPermission` with a locator string "\*" .

### 11.10.1.9 `javax.tv.service.selection.ServiceContextPermission`

All applications shall be granted a `javax.tv.service.selection.ServiceContextPermission` with a name string "getServiceContentHandlers" and action string "own". The name string "access" shall not be granted.

### 11.10.1.10 `java.util.Locale.setDefault`

This method shall throw a security exception.

## 11.10.2 Additional Permissions for signed applications

Signed applications can additionally request to get more permissions. These permissions are requested using the permission request file (12.6, "Security policy for applications" on page 297). This section defines the mapping from the items in the permission request file to the Java Permissions that may be granted by the MHP terminal in response to the request.

### 11.10.2.1 `java.util.PropertyPermission`

For signed applications, a read permission shall be granted for all properties for unsigned applications and `dvb.persistent.root`.

### 11.10.2.2 `java.io.FilePermission`

A read permission shall be granted for the subtree under which the implementation mounts the object carousels.

When the permission request file requests the permission to access persistent storage and this is granted, a `FilePermission` that permits access to the persistent storage directory subtree is created.

When there is a persistent file credential in the permission request file and this is granted, `FilePermissions` are created as follows:

- file path = value of `dvb.persistent.root` property + filename from the credential
- action = string containing "read" and/or "write" as indicated in the credential

### 11.10.2.3 `org.dvb.net.ca.CAPermission`

When the permission request file requests the permission to communicate with a CA system and this is granted, a `CAPermission` is created as follows:

The CA system ID string from the permission request file is directly used as the first part of the string used for the `CAPermission` constructor, this is concatenated with a colon character and the list of the attribute strings based on the attributes listed as true in the permission request file.

### 11.10.2.4 `org.dvb.application.AppsControlPermission`

When the permission request file requests the permission to have additional permissions to control the lifecycle of applications and this is granted, an `AppsControlPermission` is created.

### 11.10.2.5 org.dvb.net.rc.RCPermission

When the permission request file requests the permission to communicate through the return channel and this is granted, an `RCPermission` is created as follows:

- for the default ISP item, the `RCPermission` is created with "target:default" string.
- for items with phone numbers in them, the string is "target:" + the phone number prefix in the permission request file + "\*".

On `org.dvb.net.rc.ConnectionRCInterface`, the method `getCurrentTarget` shall always throw a `SecurityException`. The method `setTarget` shall throw a security exception where the application doesn't have the permission to use the target specified. The method `setTargetToDefault` shall throw a security exception where the application doesn't have either "target:default" or "target:\*" permissions.

### 11.10.2.6 org.dvb.net.tuning.TunerPermission

When the permission request file requests the permission to access the Tuning API and this is granted, an `TunerPermission` is created.

### 11.10.2.7 javax.tv.service.selection.SelectPermission

When the permission request file requests the permission to perform service selection and this is granted, a `SelectPermission` is created with a locator "\*" and action string "\*" and a `ServiceContextPermission` is created with a name "\*" and action string "\*".

### 11.10.2.8 org.dvb.user.UserPreferencePermission

When the permission request file requests the permission to read and/or write user preferences and this is granted, a `UserPreferencePermission` is created as follows:

- when the permission request file includes "true" for the "read" attribute and this is granted, a `UserPreferencePermission` is created with the string "read"
- when the permission request file includes "true" for the "write" attribute and this is granted, a `UserPreferencePermission` is created with the string "write"

### 11.10.2.9 java.net.SocketPermission

When the permission request file requests the permission to communicate with remote hosts and this is granted, `SocketPermissions` are created with the host and action as indicated in the permission request file.

These permissions shall not be granted in MHP terminals where all return channels are represented by instances of `org.dvb.net.rc.ConnectionRCInterface` (i.e. where all return channels are connection oriented) unless an instance of `org.dvb.net.rc.RCPermission` is also granted.

### 11.10.2.10 org.dvb.media.DripFeedPermission

When the permission request file requests the permission to use the drip feed feature and this is granted, a `DripFeedPermission` shall be created.

### 11.10.2.11 org.dvb.application.storage.ApplicationStoragePermission

When the permission request file requests the permission to store applications and this is granted, a `StoredApplicationPermission` is created. This permission class is defined in Annex AG, "(normative): Stored application APIs" on page 1347.

### 11.10.2.12 org.dvb.smartcard.SmartCardPermission

When the permission request file requests the permission to access the smart card API and this is granted, a `SmartCardPermission` is created. This permission class is defined in Annex AI, "(normative): Non-CA Smartcard APIs" on page 1396 of this specification.

### 11.10.2.13 ServiceContextPermission

All applications which are granted a SelectPermission (see 11.10.2.7, "javax.tv.service.selection.SelectPermission" on page 274) shall be granted instances of `javax.tv.service.selection.ServiceContextPermission` with names:

- "create"
- "destroy"
- "stop"

all of these to have the actions string "own".

## 11.11 Content referencing

The following mapping shall be used between the types of locator defined in table 135, "Addressable entities, locators and their text representation" on page 358 and the DVB-J methods defined in this chapter. It lists the Java methods & constructors which accept or return (as defined by their method signature) instances of `org.davic.net.Locator`, `javax.tv.locator.Locator`, `javax.media.MediaLocator` or their sub-classes. The external form of these locators shall be the text representation defined in the table 135, "Addressable entities, locators and their text representation" on page 358 for the corresponding entity being referenced. Where the same method is listed as accepting multiple forms of locator, then it is required to accept all forms listed in this section.

Where a method listed below is defined (in its specification) to check its input then it shall only accept the forms of locator listed below as being valid for that method from among those defined in this specification. Other forms of locator from among those defined in this specification shall be rejected as specified for the method concerned. If a method does not specify a means of rejecting inappropriate locators then it shall fail silently apart from Exceptions and Events which do not check their input and where it is the responsibility of the platform to use correct locators when constructing them. This specification does not prevent methods accepting other forms of locator which are not defined in this specification.

### 11.11.1 Transport stream

The following methods used in this specification shall accept or return instances of Objects which describe an MPEG transport stream. Methods which accept a locator for a transport stream as an input parameter shall also accept all other DVB locators which include the information to identify a transport stream. Any excess information shall be silently discarded by such methods.

- `javax.tv.service.transport.TransportStream.getLocator()`
- `javax.tv.service.transport.TransportStreamCollection.retrieveTransportStream()`
- `org.davic.net.tuning.StreamTable.getTransportStreams()`
- `org.davic.net.tuning.NetworkInterfaceController.tune()`
- `org.davic.net.tuning.NetworkInterface.getLocator()`
- `org.davic.net.tuning.NetworkInterfaceController.reserveFor()`
- `org.davic.net.tuning.StreamTable.listTransportStreams()`
- `org.dvb.si.SITransportStream.getDvbLocator()`

### 11.11.2 Network

The following methods used in this specification shall accept or return instances of Objects which describe a DVB network.

- `javax.tv.service.transport.NetworkCollection.retrieveNetwork()`
- `javax.tv.service.transport.Network.getLocator()`

### 11.11.3 Bouquet

The following methods used in this specification shall accept or return instances of Objects which describe a DVB bouquet.

- `javax.tv.service.transport.BouquetCollection.retrieveBouquet()`
- `javax.tv.service.transport.Bouquet.getLocator()`

### 11.11.4 Service

#### 11.11.4.1 MPEG/DVB specific service

The following methods used in this specification shall accept or return instances of Objects which describe a DVB service. Methods which accept a locator for a MPEG/DVB service as an input parameter shall also accept all other DVB locators which include the information to identify a service. Any excess information shall be silently discarded by such methods.

- `org.davic.net.ca.CAModule.buyEntitlement()`
- `org.davic.net.ca.CAModule.queryEntitlement()`
- `org.dvb.si.SIDatabase.retrieveSIService()`
- `org.dvb.si.SIDatabase.retrievePMTService()`
- `org.dvb.dsmcc.DSMCCStream.getStreamLocator` where the method `isMPEGProgram` returns `true`
- `org.dvb.dsmcc.ServiceDomain.attach()`
- `org.dvb.dsmcc.ServiceDomain.getLocator()`
- `org.dvb.si.PMTService.getDvbLocator()`
- `org.dvb.si.SIBouquet.getSIServiceLocators()`
- `org.dvb.si.SIService.getDvbLocator()`
- `org.davic.net.ca.TuneRequestEvent` - constructor
- `org.davic.net.ca.TuneRequestEvent.getLocator()`
- `org.dvb.application.AppAttributes.getServiceLocator()` where the service is a MPEG / DVB service
- `org.dvb.dsmcc.ServiceXFRReference` - constructor where the service is a MPEG / DVB service
- `org.dvb.dsmcc.ServiceXFRReference.getLocator()` - where the service is a MPEG / DVB service



### 11.11.4.2 Generic Service

The following methods used in this specification shall accept or return instances of Objects which reference generic services. These methods are also required to accept or return the same locators as in the previous section.

- `javax.tv.service.navigation.LocatorFilter` - constructor
- `javax.tv.service.navigation.LocatorFilter.getFilterValue()`
- `javax.tv.service.SIManager.getService()`
- `javax.tv.service.navigation.ServiceDetails.getLocator()`
- `javax.tv.service.Service.getLocator()`
- `javax.tv.service.SIManager.retrieveServiceDetails()`
- `javax.tv.service.navigation.ServiceList.findService()`
- `org.dvb.application.AppAttributes.getServiceLocator()` - where the service is not a MPEG / DVB specific one
- `org.dvb.dsmcc.ServiceXFRReference` - constructor where the service is not a MPEG / DVB specific one
- `org.dvb.dsmcc.ServiceXFRReference.getLocator()` where the service is not a MPEG / DVB specific one
- `org.dvb.dsmcc.ServiceXFRException` - constructor
- `org.davic.media.MediaLocator` - constructor
- `javax.media.MediaLocator` - constructor
- `javax.media.Manager.createPlayer(MediaLocator)`
- `javax.media.Manager.createDataSource(MediaLocator)`
- Constructor of various JMF events consumes JMF MediaLocators:  
`org.dvb.media.CAStopEvent, org.dvb.media.PresentationChangedEvent, org.dvb.media.ServiceRemovedEvent, org.dvb.media.NoComponentSelectedEvent`
- Various JMF events return JMF MediaLocators which were passed into their constructors:  
`org.dvb.media.CAStopEvent, org.dvb.media.PresentationChangedEvent, org.dvb.media.ServiceRemovedEvent, org.dvb.media.NoComponentSelectedEvent`
- `StoredApplicationService.getLocator` - return value
- `InternetClient.getServiceContentLocators` - return value

### 11.11.5 DVB Event

The following methods used in this specification shall accept or return instances of Objects which describe a DVB event.

- `javax.tv.service.guide.ProgramEvent.getLocator()`
- `javax.tv.service.SIManager.retrieveProgramEvent()`
- `org.davic.net.ca.CAModule.buyEntitlement()`
- `org.davic.net.ca.CAModule.queryEntitlement()`
- `javax.tv.service.guide.ProgramSchedule.retrieveProgramEvent()`
- `org.dvb.si.SIEvent.getDvbLocator()`

### 11.11.6 MPEG elementary stream

The following methods used in this specification shall accept or return instances of Objects which describe a MPEG elementary stream. Methods below which accept as an input parameter an array of locators shall also accept DVB locators including multiple component tags.

- `javax.tv.net.InterfaceMap.getLocalAddress()`
- `javax.tv.service.selection.InvalidServiceComponentException` - constructor
- `javax.tv.media.MediaSelectControl` - all methods accepting or returning instances of `javax.tv.locator.Locator`
- `javax.tv.media.MediaSelectEvent` & subclasses - constructor
- `javax.tv.media.MediaSelectPermission` - constructor
- `javax.tv.service.selection.ServiceContext.select()`
- `org.dvb.si.SIDatabase.retrievePMTElementaryStreams()`
- `org.dvb.si.PMTElementaryStream.getDvbLocator()`
- `org.dvb.dsmcc.ServiceDomain.attach()`
- `org.dvb.dsmcc.ServiceDomain.getLocator()`
- `javax.tv.media.MediaSelectControl.getCurrentSelection()`
- `javax.tv.service.navigation.ServiceComponent.getLocator()`
- `javax.tv.media.MediaSelectEvent.getSelection()` (also subclasses)
- `org.davic.net.ca.DescramblingStoppedEvent.getServiceLocator()`
- `org.davic.net.ca.DescramblingStartedEvent.getServiceLocator()`
- `org.davic.media.MediaLocator` - constructor - shall also accept multiple component tag "dvb:" locator
- `javax.media.MediaLocator` - constructor - shall also accept multiple component tag "dvb:" locator
- `javax.media.manager.createPlayer(MediaLocator)` - shall also accept multiple component tag "dvb:" locator
- `javax.tv.service.selection.InvalidServiceComponentException.getInvalidServiceComponent()`
- `javax.tv.service.selection.ServiceContentHandler.getServiceContentLocators()`

### 11.11.7 File

The following methods used in this specification shall accept or return locators which reference files.

- `org.davic.media.MediaLocator` -constructor - for audio files intended to be played from memory
- `javax.media.MediaLocator` - constructor - for audio files intended to be played from memory
- `javax.media.Manager.createPlayer(MediaLocator)` - for audio files intended to be played from memory
- `javax.media.Manager.createDataSource(MediaLocator)` - for audio files intended to be played from memory
- `org.dvb.dsmcc.ServiceDomain.getURL(Locator)` - instances of "dvb:" locator including `dvb_abs_path`

Apart from the above, all file references shall be encapsulated in instances of `java.net.URL`.

### 11.11.8 Directory

The following method shall return an instance of the "dvb:" locator referencing a directory.

- `org.dvb.application.AppIcon.getLocator()`

### 11.11.9 Drip feed decoder

The following methods used in this specification shall accept or return locators which reference the "drip feed" decoder.

- `javax.media.MediaLocator` - constructor
- `javax.media.Manager.createDataSource(MediaLocator)`

### 11.11.10 Irrelevant

The following methods used in this specification which accept Locators according to their signature have no requirement to have locator types specified for them in this specification:

- `javax.tv.service.ReadPermission`, all applications shall have this permission with "\*", see 11.10.1.8, "`javax.tv.service.ReadPermission`" on page 273.
- `javax.tv.service.selection.SelectPermission`, applications shall either have this permission with "\*" or not to have it at all, see 11.10.2.7, "`javax.tv.service.selection.SelectPermission`" on page 274.

### 11.11.11 Methods working on many Locator types

The following methods used in this specification work on many locator types. The locator types which each method is required to support are listed for each of the methods concerned.

- `javax.tv.locator.LocatorFactory.transformLocator` - transforms a transport independent locator into a transport independent one  
 required to accept instances of `org.davic.net.dvb.DvbLocator`  
 required to return instances of `org.davic.net.dvb.DvbNetworkBoundLocator`
- `javax.tv.locator.LocatorFactory.createLocator` - creates a locator from a string  
 required to accept valid 'dvb' URLs (see 14.1, "`Namespace mapping (DVB Locator)`" on page 353) and return corresponding instances of `org.davic.net.dvb.DVBLocator`
- `javax.tv.service.SIManager.registerInterest` - accepts a locator referencing one or more SIElements as input
- `javax.tv.service.SIManager.retrieveSIElement` - accepts a locator referencing one or more SIElements as input  
 Both these methods are required to accept locators referencing: -Bouquet, Network, Event, ElementaryStream, Service, TransportStream
- `javax.tv.service.SIElement.getLocator`  
 returns a locator for "this SIElement" as specified by the JavaTV specified sub-interfaces, no other SIElements exist

### 11.11.12 Support for the HTTP protocol in DVB-J

In MHP terminals where an HTTP protocol (6.3.7.1, "HTTP 1.1" on page 58 or 6.3.7.2, "MHP profile of HTTP 1.0" on page 58) is supported, the following classes and methods shall support the HTTP protocol concerned. In MHP terminals where the HTTPS protocol (6.3.7.3, "HTTPS" on page 59) is supported, the following class and methods shall support that protocol.

- The constructor for `javax.media.MediaLocator` - for referencing audio files intended to be played from memory
- methods on `javax.media.Manager` accepting `javax.media.MediaLocator` as input parameters - for constructing JMF players for audio files intended to be played from memory
- The classes & methods in the "java.net" package
- Methods in this specification which accept instances of `java.net.URL` are required to accept instances which encapsulate an "http" URL and behave according to their specification. - e.g. `Toolkit.getImage(java.net.URL)`
- On MHP terminals supporting applications downloaded over the interaction channel as defined in 9.7, "Services and applications not related to conventional DVB services" on page 204, the method `LocatorFactory.createLocator(String)` shall additionally accept Strings containing URLs using the "http" and "https" protocols as being valid and return a corresponding `Locator`. This method shall only validate the string to the extent that this is possible without network access.
- On MHP terminals supporting applications downloaded over the interaction channel as defined in 9.7, "Services and applications not related to conventional DVB services" on page 204, the method `SIManager.getService(Locator)` shall accept such `Locators` as being valid and return a corresponding `javax.tv.service.Service`. This method shall only validate the locator to the extent that this is possible without network access.

When HTTP URLs are used with instances of `DVBClassLoader` to load DVB-J classes over the interaction channel in a signed application, the requirements of the MHP security model shall be complied with before a class is allowed to be successfully loaded from such a URL.

## 11.12 Stand-alone applications

### 11.12.1 Common behaviour

The extended MHP application model allows several forms of applications to execute in a stand-alone mode independent of any broadcast services. The environment of these stand-alone applications shall be as follows:

- Instances of `org.dvb.si.SIDatabase` may not return any SI objects if the corresponding network interface isn't tuned to anything. It is implementation dependent whether some network interfaces are left tuned to some previously used transport stream.

For DVB-J applications, the behaviour defined in 9.7.4, "Common behaviour" on page 206 shall mean the following:

- The method `ServiceContext.getServiceContentHandlers` shall not return a `ServiceContentHandler` for any such existing video or audio for a stand-alone stored application.
- A stand-alone application creating and starting a JMF player shall receive a higher priority for resources than the presentation of any previously existing video and audio.

### 11.12.2 Stored services

The MHP specification supports two forms of stored applications as defined in 9.1.9, "Broadcast service related stored applications" on page 186 and 9.7.2, "Stored services" on page 204.

#### 11.12.2.1 Stored application APIs

This is defined in Annex AG, "(normative): Stored application APIs" on page 1347 of this specification.

#### 11.12.2.2 Modified behaviour of MHP 1.0 APIs

The following APIs defined in MHP 1.0 shall have extended semantics to support listing and launching of stored services as defined in 9.7.2, "Stored services" on page 204. The modifications to these APIs are as follows:-

- The class `javax.tv.service.navigation.ServiceTypeFilter` when used with a `ServiceType "org.dvb.application.StoredApplicationService.STORED_APPLICATION_SERVICE"` shall return a `ServiceList` containing all currently installed stored services when passed to `SIManager.filterServices`.

NOTE: Implementations may choose to expose other installed services (e.g. manufacturer resident ones) through this API.

- Instances of `javax.tv.service.Service` returned from such a `ServiceList` shall be instances of `org.dvb.application.StoredApplicationService` with the modified semantics as defined for that interface.
- The method `ServiceContext.select(Service selection)` shall accept instances of `org.dvb.application.StoredApplicationService` and attempt to start the stored application service concerned subject to the MHP security model.
- The file system APIs shall see a file system comprised of the files defined in the application description file as needing to be stored for this application. Calling `new DSMCCObject(".")` or `new java.io.File(".")` shall instantiate the directory object that refers to the outermost (top level) directory listed in the application description file (see 10.14.3, "Application description file" on page 239) when the application was stored. Other file system behaviour shall be as defined in 9.7.2, "Stored services" on page 204.
- The `org.dvb.dsmcc` API shall work for access to stored files. Asynchronous loading can be expected to complete immediately. `ObjectChangeEvents` shall not be generated. Pre-fetching shall be always supported for stored files. The `unload()` method shall not be considered as removing stored files from where they are stored.

## 11.13 Support for DVB-HTML

### 11.13.1 Document object model APIs

#### 11.13.1.1 Framework

The required document object model APIs are defined in the following:

- From [Document Object Model \(DOM\) Level 2 Core Specification \[82\]](#), that part of Annex D, "Java Language Binding" corresponding to the "Fundamental Interfaces" as specified in section 1.2 of that specification.
- From [Document Object Model \(DOM\) Level 2 Events Specification \[84\]](#), Annex B, "Java Language Binding", only those parts corresponding to the interfaces listed as required in 8.11.1.1, "Fundamental interfaces" on page 128 and 8.11.1.2, "Event interfaces" on page 128.
- From [Document Object Model \(DOM\) Level 2 Views Specification \[86\]](#), Annex B, "Java Language Binding".

#### 11.13.1.2 DVB defined extensions

The DVB defined DOM extensions are defined in Annex AD, "(normative): Support for DVB-HTML" on page 1296 of this specification see also AD.1, "Java bindings to DVB extensions" on page 1296.

#### 11.13.1.3 Read Only Access to DOM

Where a DVB-J application only has or requests read-only access to the DOM, calling methods in the DOM APIs whose result would be to modify the DOM shall result in an `org.w3c.dom.DOMException` with exception code `NO_MODIFICATION_ALLOWED_ERR` being thrown.

### 11.13.2 Embedded Xlets in a DVB-HTML Page

Below is a list of the semantic differences in MHP APIs between DVB-J applications and xlets embedded in a DVB-HTML page of a DVB-HTML application.

- Xlets shall not use the HAVi HScene APIs to get the "primary" Container that is managed by the enclosing DVB-HTML document. The method `javax.tv.graphics.TVContainer.getRootContainer(javax.tv.xlet.XletContext)` shall be used for this purpose. The object returned shall be a subclass of `java.awt.Container` implementing the following interfaces:
  - `org.havi.ui.HComponentOrdering`
  - `org.dvb.application.inner.DVBScene`
 It may be an instance of `org.havi.ui.HScene` or a subclass of that but is not required to be either of these.
- Embedded Xlets are only guaranteed to be able to perform actions using the Document Object Model APIs when they are in the active state. The results of calling methods on `org.dvb.dom.bootstrap.DocumentFactory` when the embedded xlet is in any other state are implementation dependent. They may include a `RuntimeException` being thrown or the Document passed to `DocumentAction.run()` being null. If the state of an application changes from the active state to another state between when a method on `DocumentFactory` is called and when the `run()` method of the specified `DocumentAction` is called then the results of this are also implementation dependent.
- If the Xlet is in the active state or if the `startXlet` method is executing, and the width and height are  $> 0$ , then `getRootContainer()` shall return a valid `Container` instance. If the width or height are  $\leq 0$ , `getRootContainer` shall return null. In all other cases, situations and xlet states, the value returned is implementation dependent.

Any "param" elements of the `<object>` used to define an embedded xlet shall be accessible to the embedded xlet. The sequence of these arguments is mapped to `XletContext.ARGs` and shall be available to the Xlet, in the form of an array of Strings, by calling:

```
javax.tv.xlet.XletContext.getXletProperty(XletContext.ARGs)
```

## 11.14 Internet access

### 11.14.1 Internet client control APIs

These are defined in the `org.dvb.internet` package (see [AH, "\(normative\): Internet client APIs" on page 1365](#)).

Internet clients accessible to MHP applications shall appear in the list of services maintained by the `SIManager` which is returned by calls to the method `SIManager.filterServices`. For each accessible internet client, the corresponding sub-class of `InternetClientService` shall be returned from calls to that method both when passed an instance of `ServiceTypeFilter` constructed with the type `InternetServiceType.INTERNET_CLIENT` and when passed null.

The method `ServiceContext.select(Service selection)` shall accept instances of `InternetClientService` and its sub-classes returned by the mechanism described above as being valid input.

### 11.14.2 Internet applet support

Internet applet support is defined as follows.

#### 11.14.2.1 HTML tags

Internet applets shall be supported using the "<applet>" and "<object>" tag with the semantics defined in [HTML 4 \[90\]](#).

#### 11.14.2.2 Common MHP features

The following features of the DVB-J platform shall be supported for internet applets in the same way as they are supported for DVB-J applications.

- [The Virtual Machine see 11.1 on page 241](#)
- [Approach to Subsetting \(see 11.2.2 on page 241\)](#)
- [Unloading \(see 11.2.4 on page 242\)](#)
- [java.lang.reflect package \(see 11.3.1.2 on page 244\)](#)
- [java.util \(see 11.3.1.3 on page 244\)](#)
- [java.util.zip \(see 11.3.1.4 on page 244\)](#)
- [java.io \(see 11.3.1.5 on page 244\)](#)
- [java.net \(see 11.3.1.6 on page 244 and 11.5.3, "Support for IP over the Return Channel" on page 258, 11.11.12, "Support for the HTTP protocol in DVB-J" on page 280\)](#)
- [java.beans \(see 11.3.1.7 on page 245\)](#)
- [java.math \(see 11.3.1.8 on page 245\)](#)
- [java.awt.event \(see 11.4.1.1, "The Core GUI API" on page 246\)](#)
- [java.awt.image \(see 11.4.1.1, "The Core GUI API" on page 246\)](#)
- [java.security \(see 11.8.1.1 on page 268 and 11.8.2, "APIs to Support TLS / SSL Over the Return Channel" on page 269\).](#)
- [java.security.cert \(see 11.8.1.2 on page 269\)](#)

#### 11.14.2.3 Extended or modified common MHP features

The following features of the DVB-J platform shall be supported for internet applets in the same way as they are supported for DVB-J applications except as modified below.

- [Basic Considerations \(see 11.2.1 on page 241\)](#)

The requirements on DVB-J applications do not apply to internet applets.

- [java.lang package](#) (see 11.3.1.1 on page 243)

The full set of system properties required to be supported for `System.getProperty()` and `System.getProperties` by [JAE 1.1.8 API \[31\]](#) are required to be supported.

- [java.text](#) (see 11.3.1.9 on page 245)

The complete `java.text` package as defined in [JAE 1.1.8 API \[31\]](#) shall be supported.

#### 11.14.2.4 The java.awt package

The `java.awt` package shall be supported as defined in section 5.2 of [PersonalJAE \[36\]](#) as modified below.

- Support for deprecated methods is required in this package and its sub-packages.
- The following class of the `java.awt` package is not required:

`FileDialog`

- The following method of the `java.awt.Component` class is not required:

`isDoubleBuffered()`

- The following methods of the `java.awt.Toolkit` class are not required:

`getMenuShortcutKeyMask()`, `getPrintJob()`, `getSystemEventQueueImpl()`, `loadSystemColors(int[])`

- Where [PersonalJAE \[36\]](#) has optional behaviour or defines multiple levels of support, the following minimums shall be supported.

Dialog: Minimal support

Frame: Minimal support

Window: Minimal support

`Component.setCursor()`: The specified cursor may be ignored

`Dialog.setResizable()`: Unsupported

`PopupMenu.add(MenuItem)`: Nested `PopupMenu`s supported

`Graphics.setXORMode()`: Supported

Scrolling controls: As defined in section 5.2.1 of [PersonalJAE \[36\]](#)

Menu option: Supported (see table 2 of 5.2 of [PersonalJAE \[36\]](#))

Scrollbars: Supported (see table 2 of 5.2 of [PersonalJAE \[36\]](#))

- Where the PJAE specification indicates that `com.sun.lang.UnsupportedOperationException` shall be thrown to indicate the absence of an optional operation, a subclass of `java.lang.RuntimeException` shall be permitted instead.

#### 11.14.2.5 Additional features for applet support.

The following additional features shall be supported for internet applets.

- The `java.applet` package as defined in [JAE 1.1.8 API \[31\]](#).

This specification does not require support for any specific audio formats for use with the `java.applet.AudioClip` feature. Attempting to play Audio Clips where audio is not supported shall fail silently.

- The `java.awt.datatransfer` package as defined in [JAE 1.1.8 API \[31\]](#)

#### 11.14.2.6 Security

This specification does not require support for signed applets or for the code signing mechanism and APIs as defined in [PersonalJAE \[36\]](#).



Applets shall have at least the following permissions.

- `java.util.PropertyPermission` to "read" all properties defined for the `java.lang.System.getProperties()` method.
- `java.net.SocketPermission` to "connect" to all ports on the server from which the applet was originally downloaded.

---

# 12 Security

## 12.1 Introduction

This section covers the following areas of security:

- Authentication of applications
- Security policies for applications
- Authentication and privacy of the return channel communications
- Certificate management

### 12.1.1 Overview of the security framework for applications

The security framework enables a receiver to authenticate the source of application code or other files. In the case of application code files, the authentication advises the receiver what access rights should be granted to an application for sensitive resources, see 12.6, "Security policy for applications" on page 297 for more detail.

The system uses 3 different security messages:

- Cryptographic hash codes

This provides a summary of a quantity of data - typically a subset of the total set of data under consideration.

- Signatures

These deliver a master hash code (computed over all of the appropriate data) that has been "signed" by an authorising organisation. The signing process securely associates the master hash code with the signatory. The hash code process shows that the data has not been tampered with since it was signed by the signatory.

- Certificates

These provide a "chain of trust" from the authorising organisation up to some trusted third party (the root certificate authority) that is well known to the receiver.

The messages are delivered within files of the file system so this authentication scheme is applicable to any hierarchical file system whether operating over the broadcast or return channels.

Applications that are signed shall be identified with an [application\\_id](#) from the signed applications range (see table 78, "Value ranges for application\_id" on page 214). Applications that are unsigned shall be identified with an [application\\_id](#) from the unsigned applications range. An application with an [application\\_id](#) from the signed applications range but that is not signed is considered to have failed authentication. An application with an [application\\_id](#) from the unsigned applications range is treated as unsigned even if the files might be transmitted with signatures.

### 12.1.2 Overview of return channel security

In this version of the specification general purpose protocols and a standard cryptographic suite derived from internet standards are used.

## 12.2 Authentication of applications

### 12.2.1 Overview of authentication messages

Three different message types are used: "Hash codes", "Signatures" & "Certificates". Each message is placed in a file. The placement of the files depends on their function and is specified under the appropriate headings under 12.4, "Detail of application authentication messages" on page 289.

### 12.2.1.1 Hash codes

This specification describes application of hash codes to the following types of information:

- Files
- Directories

The hash computation considers the content and attributes of the objects rather than transport specific information. As a result, the authentication is independent of the underlying transport protocol.

In the case of a directory the hash value depends on the hash values of the objects bound to it, and so provides a hash of all of the objects to be authenticated in the "tree" below it.

### 12.2.1.2 Signatures

The data authenticated is a hierarchical file system (for example DSM-CC OC). The root of an authenticated "tree" carries one or more signatures. This allows one or more organisations to sign a set of information.

The root of the authenticated "tree" can be the root directory of the file system or the "top" directory of a "subtree".

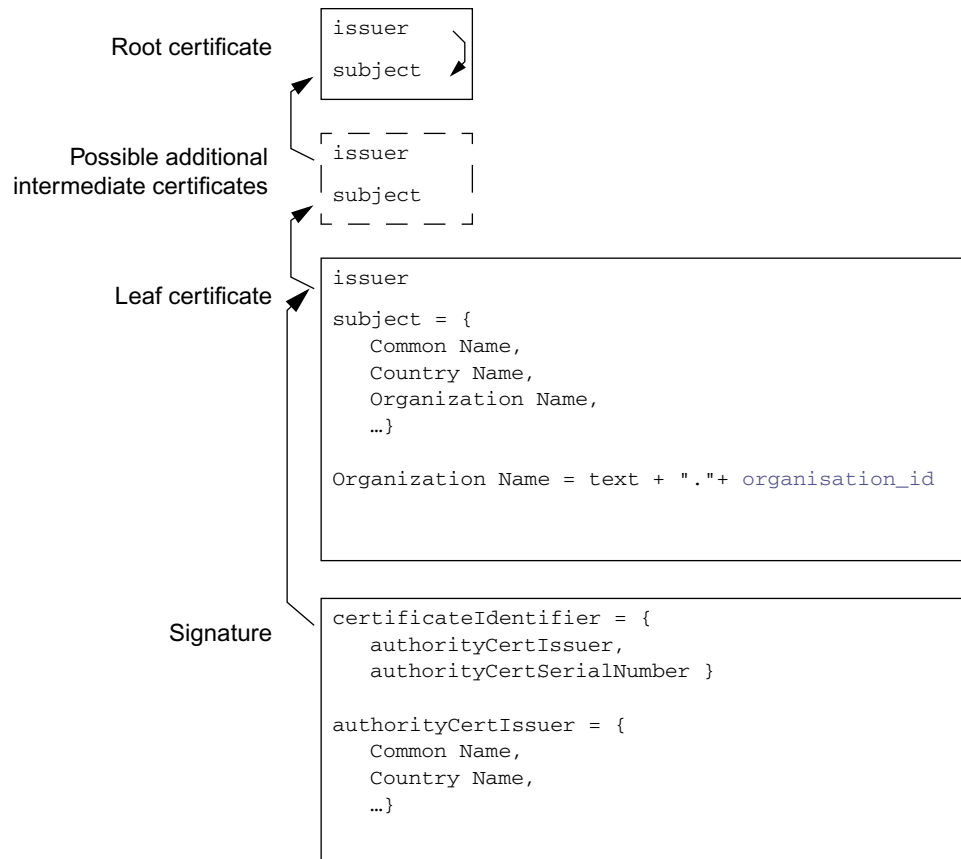
The signature:

- references a certificate containing the public key required to decode the signature
- identifies the hash algorithm used
- and the value of the signature

### 12.2.1.3 Certificates

The certificate provides a public key that can be used to decode a hash code contained in a signature and so enable a tree/subtree to be verified. The certificate itself is signed by a higher certification authority.

To correctly authenticate a subtree there must be a valid "chain" of certificates from the signature to a root certificate as is illustrated in figure 19.



**Figure 19 : Certificate chain illustrated**

#### 12.2.1.4 Authentication of hierarchical file systems

The solution here is based on authentication of a hierarchical structure of objects. Hashcodes are computed systematically and accumulatively across some or all of the objects in the hierarchy. A signature at the top of the hierarchy identifies the source of the objects.

The framework provides a flexible and non-time consuming method enabling the authentication of subtrees of a file system with a single signature. Since checking signature is far more time-consuming than checking hashcode values, this mechanism is more efficient than signing each object of a subtree.

Further, only the objects that are loaded need real-time hashcode checking.

This mechanism does NOT mandate that the whole subtree is authenticated.

**NOTE:** The broadcaster can choose which files in the file system are authenticated. For example code files might be authenticated and asset files might be left without authentication.

Finally, this framework embraces key distribution by specifying a certificate mechanism, the aim of this is to certify that the key used to compute the signature is valid and used by a certified service provider. See 12.8, "Procedures for application certificates and signatures" on page 314.

## 12.3 Message transport

The security messages are transported in files.

In no cases shall a service transfer be required to access the file content. In the case that the file system is an object carousel this means that the IOR for the security files shall always use a BIOP profile body and never a Lite options profile body.

## 12.4 Detail of application authentication messages

Three data structures are defined for communicating authentication information:

- "HashFile" on page 289
- "SignatureFile" on page 291
- "CertificateFile" on page 292

These are placed in files in the file system. The location of the file depends on its function.

### 12.4.1 HashFile

#### 12.4.1.1 Description

The [HashFile](#) lists all of the elements of the current directory except itself and the possible signature files. Those elements to be authenticated are associated with hashcodes. The syntax of the [HashFile](#) is shown in table 119.

**Table 119 : Syntax of the Hashfile**

Syntax	Num. Bits	Format
<pre> Hashfile () {   digest_count   for( i=0; i&lt;digest_count; i++ ) {     digest_type     name_count     for( j=0; j&lt;name_count; j++ ) {       name_length       for( k=0; k&lt;name_length; k++ )         name_byte     }   }   for( j=0; j&lt;digest_length; j++ ) {     digest_byte   } } </pre>	<p>16</p> <p>8</p> <p>16</p> <p>8</p> <p>8</p> <p>8</p>	<p>uimsbf</p> <p>uimsbf</p> <p>uimsbf</p> <p>uimsbf</p> <p>bslbf</p> <p>bslbf</p>
Other data may follow but can be ignored by implementations conforming to this profile.		

**digest\_count:** This 16 bit value identifies the number of digest values in this hash file.

**digest\_type:** This 8 bit value identifies the digest algorithm, if any, used for the associated objects. Table 120 lists the allowed values for this field.

**Table 120 : Values of digest\_type**

value	digest len.	algorithm
0	0	Non authenticated
1	16	MD-5 as defined in RFC 1321 [38]
2	20	SHA-1 as defined in FIPS-180-1 [62]
Other values		Reserved

**name\_count:** This 16 bit value identifies the number of object names associated with the digest value. The value of this field shall be greater than zero.

**name\_length:** This 8 bit value identifies the number of bytes in the object name.

**name\_byte:** This 8 bit value holds one byte of the object name.

Each name shall be the name of an object in the directory that contains the [HashFile](#). So, file names are the names of files in the directory and directory names are the names of direct sub-directories of the directory. No path information shall be included in the name.

The names carried by this field are binary identical to the payload part of names in the file system. So, any name matching process can be binary and ignorant of character encoding, letter case etc. Also, terminating null characters are not considered to be part of the file name.

**digest\_length:** This integer value gives the number of bytes in each digest value. It depends upon the digest type as tabulated in table 120.

NOTE: Non-authenticated objects have a zero length digest.

**digest\_byte:** This 8 bit value holds one byte of the digest value. See 12.4.1.3, "Digest value computation rules" on page 290.

### 12.4.1.2 [HashFile](#) location and naming conventions

An application comprises files containing data that can be spread across various directories and is contained within a subtree of the file hierarchy. A [HashFile](#) will be put in each directory containing objects that need to be authenticated.

The name of the [HashFile](#) shall be:

```
"dwb.hashfile"
```

There shall only be one instance of [HashFile](#) per directory that contains authenticated resources.

See 12.7, "Example of creating an application that can be authenticated" on page 312.

### 12.4.1.3 Digest value computation rules

The digest value is computed over the objects named in the [HashFile](#) in the order listed in the [HashFile](#). The length of the list of objects associated with each digest value may be one or more.

Each list of objects may contain an arbitrary mix of different object types (e.g. a mixture of file and directory names). The digest value is computed by first initialising the digest algorithm in an algorithm specific way and then applying the relevant data for each object to the algorithm in order. The relevant data for each object depends on its type and is specified in table 121.

**Table 121 : Data required for digest value computation**

Object type	Relevant data
File	The entire content of the file
Directory	The content of the <a href="#">HashFile</a> of the named directory

### 12.4.1.4 Special authentication rules

- a) For objects which are directories, if the [digest\\_type](#) is non-zero there shall be a [HashFile](#) in the sub-directory listed. If the [HashFile](#) is absent then the authentication fails.
- b) Each [HashFile](#) shall provide a complete list of all the objects named in the directory except itself and any possible signature files mentioning each name exactly one time. The behaviour varies depending on the ability of the transport protocol to support directory listing.
  - On transport protocols supporting the listing of files in a directory (e.g. Object Carousel), the authentication shall fail if the set of objects listed in the [HashFile](#) is different to the set of objects in the directory.
  - On transport protocols that don't support the listing of files in a directory (e.g. HTTP), the set of names in the [HashFile](#) acts as a filter for the set of objects that can be accessed. Attempts to access a file not named in the [HashFile](#) shall fail (that is behave as if the file is not available). The file system for loading applications from the interaction channel via HTTP is a case of such a file system, see 6.4.1.4 on page 61.

These rules apply regardless of the value of [digest\\_type](#) associated with the object.

- c) If the `digest_count` is equal to 0, the file hashfile shall be ignored. Every entry in the directory will be non authenticated.
- d) If a `name_length` is equal to zero (or if a name is not found in the directory containing the `HashFile`), all the names associated with the current digest value shall be considered as incorrectly authenticated. It means the authentication checking will fail for these entries.
- e) If the object is a Stream or StreamEvent then the `digest_type` shall be zero.
- f) Objects associated with a `digest_type` that the receiver does not support shall be treated by that receiver as if the value of `digest_type` was zero (not authenticated).
- g) There is no requirement for certificates files to have a `digest_type` other than zero (not authenticated). However, if they do have a non-zero `digest_type` they don't receive exceptional treatment.

## 12.4.2 SignatureFile

### 12.4.2.1 Description

The `SignatureFile` is a File containing one digital signature. It contains the following ASN.1 DER structure:

```
Signature ::= SEQUENCE {
    certificateIdentifier           AuthorityKeyIdentifier,
    hashSignatureAlgorithm         HashAlgorithmIdentifier,
    signatureValue                 BIT STRING }
```

**certificateIdentifier** : As defined in the [ITU-T X.509 \[54\]](#) extension for the AuthorityKeyIdentifier field. It identifies the certificate that carries the certified public key that is used to check the signature.

```
AuthorityKeyIdentifier ::= SEQUENCE {
    keyIdentifier                 [0] KeyIdentifier OPTIONAL,
    authorityCertIssuer           [1] GeneralNames OPTIONAL,
    authorityCertSerialNumber     [2] CertificateSerialNumber OPTIONAL }
```

Implementations are not required to use the possibly present `keyIdentifier` element of the `AuthorityKeyIdentifier`. The `AuthorityKeyIdentifier` structure shall contain both the `authorityCertIssuer` and `authorityCertSerialNumber` elements.

The `authorityCertIssuer` shall contain the field "directoryName", this field shall be equal to the `issuerName` of the certificate that carries the public key used to check the signature.

**hashSignatureAlgorithm**: this field identifies the hash algorithm that is used. Note that the encryption algorithm used to compute the signature is already described in the `SubjectKeyInfo` field of the certificate that certifies this key, and thus that only the identification of the hash algorithm is needed. The supported algorithms are MD5 and SHA-1.

```
md5 OBJECT IDENTIFIER ::=
{ iso(1) member-body(2) US(840) rsadsi(113549)
  digestAlgorithm(2) 5 }
```

```
sha-1 OBJECT IDENTIFIER ::=
{ iso(1) identified-organization(3) oiw(14) secsig(3)
  algorithm(2) 26 }
```

**signatureValue**: See "signatureValue" on page 323.

### 12.4.2.2 SignatureFile location and naming conventions

The `SignatureFile` is located in the root directory of the subtree that it signs. There can be several `SignatureFiles`, as there can be several entities that sign the structure. See 12.4.4, "Integration" on page 293.

By convention, the name of a `SignatureFile` is:

```
"dvb.signaturefile."<x>
```

where the <x> is a textual representation of an integer decimal number without leading zeroes. The range of values represented in any single directory shall start with 1 and increment in steps of 1. The first unused integer value in the ascending sequence indicates the end of the range.

The purpose of this x is to allow the hash file of an authenticated subtree to be signed by more than one entity. It is equal to the x value in the file name of the corresponding certificate file. See 12.4.3.5, "CertificateFile location and naming conventions" on page 293 and 12.4.4, "Integration" on page 293.

### 12.4.2.3 Supported algorithms

Signing data is a two-step process:

- first a hash is computed over the data.
- the resulting hashvalue is then encrypted using an encryption algorithm.

As indicated in 12.4.1 "HashFile" this specification defines two possible hash algorithms: MD5 and SHA-1.

The encryption algorithm used to compute the signature is indicated in the certificate that carries this key.

### 12.4.2.4 Signature computation rules.

The hash is computed over the content of the HashFile contained in this root directory.

NOTE: This is the same principle as for the classical hash computation described in "Digest value computation rules" on page 290.

### 12.4.2.5 Authentication rules

See signatureValue on page 291.

## 12.4.3 CertificateFile

### 12.4.3.1 Description

The CertificateFile contains all of the certificates in the certificate chain up to, and including, the root certificate. The leaf certificate is placed first in the file. The last certificate in the file is the root certificate. The root certificate is included in this file only for consistency. How the MHP terminal determines its policy related to this root CA is implementation dependent. The encoding of the certificate is defined in ITU-T X.509 [54]. Below is defined the profile of ITU-T X.509 [54] for use in authenticating MHP applications. This profile is based on RFC 2459 [58].

The syntax of the CertificateFile is shown in table 122.

**Table 122 : Syntax of the CertificateFile**

Syntax	Num. Bits	Format
<pre> Certificatefile () {     certificate_count     for( i=0; i&lt;certificate_count; i++ ) {         certificate_length         certificate()     } } </pre>	16	uimsbf
	24	uimsbf

**certificate\_count:** This 16-bit integer carries the number of certificates in the certificate file.

**certificate\_length:** This 24-bit integer specifies the number of bytes in the certificate.

**certificate():** This field carries a single "Certificate" data structure as defined by ITU-T X.509 [54]. See 12.11.1, "Main part of the certificate" on page 323.



### 12.4.3.2 ASN.1 encoding

The basic specification of the ASN.1 DER encoding used in RFC 2459 [58] is given in ASN.1 [57]. However, RFC 2459 [58] defines some extensions which are required to implement this specification.

### 12.4.3.3 Supported algorithms

There are various algorithm identifiers in the certificate structure. The OID of the algorithm used in the `SubjectPublicKeyInfo` structure shall be RSA.

The values for `AlgorithmIdentifier` used both in the certificate structure and in the `TBSCertificate` structure that are supported in this specification are listed under "`signatureAlgorithm`" on page 294.

### 12.4.3.4 Name matching

The only allowed encoding of attributes of distinguished names shall be UTF8String.

NOTE: The use of this encoding allows name matching to be a binary comparison.

### 12.4.3.5 CertificateFile location and naming conventions

As described in 12.2.1.3, "`Certificates`" on page 287, a key can be authenticated through a "certificate chain".

A certificate chain is a hierarchy of certificates that enable the implementation to verify the validity of the key used to check a signature. In the MHP environment, the root certificate is embedded in the MHP. Hence, the file structure shall carry all of the certificate chain apart from the root certificate.

The certificate file that leads to the public key of a signature shall be placed in the same directory as that signature file.

The name of a `CertificateFile` is:

```
"dvb.certificates.x"
```

where the <x> corresponds to the x value in the file name of the signature file verified by this certificate. Hence in the root directory of an authenticated subtree there shall be one certificate file for each signature file. See 12.4.2.2, "`SignatureFile location and naming conventions`" on page 291 and 12.4.4 "`Integration`".

### 12.4.3.6 Authentication rules

Certificates are considered to be the same if they have bitwise identical contents.

## 12.4.4 Integration

Logically a file is authenticated as follows:

- a) Confirm that the file is listed in the hash file located in the same directory as the file to be authenticated.
- b) Verify that the file contents and the corresponding digest value are consistent.
- c) Recursively ascend the directory hierarchy checking that each directory is authenticated by its parent directory until a directory is found that contains one or more signature files.

Such a directory is termed the root directory of an authenticated subtree.

NOTE 1: This means that there is no requirement to progress above the root of the authenticated subtree to examine further hash files. If a such a directory has a digest type other than "Non authenticated" in its parent directory, this is only significant when verifying the hash file of the parent directory.

NOTE 2: Presence of more than one signature file enables more than one set of organisations to authenticate a subtree. The application of this is outside the scope of this specification.

- d) For a signature file locate the corresponding certificate file (where the x portion of the signature file's file name identifies the certificate file to be used).

- e) Follow the certificate chain contained within the certificate file verifying each link in the chain until the link to the root certificate is found.
- f) If the identified root certificate and all the intermediate certificates leading to it are "satisfactory", accept the files as being authenticated.

"satisfactory" depends on the policies implemented in the receiver and other constraints expressed in this specification. In particular the requirements in 11.2.3, "Class Loading" on page 241 for using the "same" leaf certificate to authenticate DVB-J class files shall be observed. For HTML applications the definition of "satisfactory" is given in 8.14, "Security of DVB-HTML applications" on page 164.

- g) Dependant on receiver policy return to step (c) and repeat for other signature files.

NOTE: The above is a logical description of the process and does not constrain implementations to perform these steps in this exact order. E.g. hash files may be verified when descending an directory hierarchy rather than ascending one.

A file system may contain several independent authenticated subtrees, each tree with its own subtree root directory.

## 12.5 Profile of X.509 certificates for authentication of applications

This section identifies how ITU-T X.509 [54] is profiled when used for authentication of broadcast MHP applications. This profile is a variation (in general a sub-set) of the internet profile defined in RFC 2459 [58]. This section identifies the differences from the profile in RFC 2459 [58]. Section 12.11, "The internet profile of X.509 (informative)" on page 322 summarises the profile in RFC 2459 [58].

### 12.5.1 signatureAlgorithm

This specification supports 2 signature algorithms: MD5 with RSA and SHA-1 with RSA.

#### 12.5.1.1 MD5 with RSA

The signature algorithm with MD5 and the RSA encryption algorithm is defined in RFC 2313 [56]. As defined in RFC 2313 [56], the ASN.1 OID used to identify this signature algorithm is:

```
md5WithRSAEncryption OBJECT IDENTIFIER ::= {
  iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 4 }
```

#### 12.5.1.2 SHA-1 with RSA

The signature algorithm with SHA-1 and the RSA encryption algorithm is implemented using the padding and encoding conventions described in RFC 2313 [56]. The message digest is computed using the SHA-1 hash algorithm. The ASN.1 object identifier used to identify this signature algorithm is:

```
sha-1WithRSAEncryption OBJECT IDENTIFIER ::= {
  iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 5 }
```

#### 12.5.1.3 parameters

For both of the 2 supported algorithms the parameters component shall be the ASN.1 type NULL.

### 12.5.2 signatureValue

The RSA signature generation process and the encoding of the result is described in detail in RFC 2313 [56].

### 12.5.3 version

The version field of the certificate shall signal v3. All implementations shall support the v3 extensions as required by 12.5.9, "Extensions" on page 296.

## 12.5.4 issuer

### 12.5.4.1 minimum requirement

For this specification at least a Common Name attribute shall be provided. The text value of the attribute shall be non-empty. It shall be suitable for direct presentation to the user.

### 12.5.4.2 certificate authority responsibility

The senior certificate authority who signs a certificate shall oversee the attribute information to ensure that the information is suitable.

## 12.5.5 validity

The only allowed format for encoding time in the `validity` field is `GeneralizedTime`.

## 12.5.6 subject

The subject field is a "distinguished name". The following requirements are specified by this specification:

- The only allowed encoding attributes of the subject is `UTF8String` (see 12.4.3.4, "Name matching" on page 293)
- The minimum set of attributes that shall be present in the subject are:
  - `commonName`
  - `countryName`
- If the certificate is a "leaf certificate" (see figure 19, "Certificate chain illustrated" on page 288) then the subject shall also contain an `organizationName`.
- When encoded the `organizationName` carries organisation specific text post fixed by the `organisation_id` of the authenticated files. This integer value is represented as a fixed length 8 character hexadecimal string (with leading zeros where required). So the `organizationName` takes the form:

```
text + "." + organisation_id
```

See 14.5, "Text encoding of application identifiers" on page 357.

This field reproduces the `organisation_id` value from the application's application identifier see 10.5.1, "Encoding" on page 214. If the values of the `organisation_ids` don't match then the application is considered to have failed authentication. Applications which fail authentication at this point shall not be started on MHP terminals.

## 12.5.7 SubjectPublic Key Info

This specification supports a single public key algorithm (RSA) for the subject public key.

The key lengths that implementation are required to support are addressed in G, "(normative): Minimum Platform Capabilities" on page 442.

### 12.5.7.1 rsaEncryption

The OID `rsaEncryption` identifies RSA public keys:

```
rsaEncryption OBJECT IDENTIFIER ::= { pkcs-1 1 }
pkcs-1 OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                                rsadsi(113549) pkcs(1) 1 }
```

The parameters field shall have ASN.1 type `NULL`.

### 12.5.7.2 subjectPublicKey

The RS `subjectPublicKey` BIT STRING shall be encoded using the ASN.1 type `RSAPublicKey`:

```

RSAPublicKey ::= SEQUENCE {
    modulus          INTEGER, -- n
    publicExponent  INTEGER -- e -- }

```

The semantics of the modulus (n) and the public exponent (e) are defined in [RFC 2313 \[56\]](#).

## 12.5.8 Unique Identifiers

X.509 defines the issuerUniqueID and subjectUniqueID extensions.

CAs conforming to this profile shall not generate certificates with unique identifiers.

MHP terminals conforming to this profile are not required to be capable of parsing unique identifiers and making comparisons.

## 12.5.9 Extensions

The following restrictions and semantics are placed on the use of certificate extensions when used to authenticate applications.

**Table 123 : Profile for standard certificate extensions**

Extension	In broadcasts	In implementations	Semantic
Authority key identifier	Opt.	Opt.	Shall not be marked critical
Subject key identifier	Opt.	Opt.	Shall not be marked critical
Key usage	Mand.	Mand.	If the keyUsage extension is marked critical, for the leaf certificate the bit digitalSignature shall be set, for other certificates the bit keyCertSign shall be set. If these bits are not set then the certificate shall be ignored by the implementation.
Private key usage period	Opt.	Opt.	Shall not be marked critical
Certificate policies	Opt.	Opt.	Shall not be marked critical
Policy mappings	Opt.	Opt.	Shall not be marked critical
Subject Alternative Name	Mand.	Opt.	Shall not be marked critical The subject name unambiguously identifies the subject. It is recommended that DVB MHP implementations can read rfc822Name (email address)
Issuer Alternative Name	Mand.	Opt.	Shall not be marked critical The issuer name unambiguously identifies the issuer. It is recommended that DVB MHP implementations can read rfc822Name (email address).
Subject Directory attributes	Opt.	Opt.	Shall not be marked critical
Basic Constraints	Opt.	Mand.	May be marked critical
Name Constraints	Opt.	Mand.	May be marked critical. DVB MHP decoders shall be able to recognise name constraints when GeneralName are either directoryName or rfc822 names.
Policy Constraints	Opt.	Opt.	Shall not be marked critical
Extended key usage field	Opt.	Opt.	Shall not be marked critical
CRL Distribution points	Opt.	Opt.	Shall not be marked critical

Table 124 : Key for table 123

Keyword	When applied to ...	
	broadcasts	receivers
Mandatory	Certificates shall include these extensions.	Receivers shall observe the semantic for this information when provided.
Optional	Certificates may or may be not include these extensions. The MHP specification does not define the use of these extensions and hence broadcasters should not use them.	Receivers can ignore these extensions if present.
Critical / not-critical	Broadcasters should not mark the receiver optional extensions as critical.	Certificates containing unrecognised critical extensions shall be considered as invalid. Receivers should recognise all the extensions that can be critical.

## 12.6 Security policy for applications

### 12.6.1 General principles

This section specifies the resource access policy for the downloaded applications. The resource access policy depends on two factors

- The access rights requested by the broadcaster through the signalling
- The access rights granted by the user.

The ultimate access rights that are granted to the applications are the intersection of the access rights requested by the broadcaster and the access rights granted by the user.

Unsigned applications have limited access to platform resources.

Unless specified elsewhere in this specification, signed applications have the same access rights as unsigned applications. An application broadcaster can request additional permissions to access specific resources by providing a signed "Permission Request File" along with the application. The syntax and semantics of the Permission Request File are defined in the following sections. The permission request file may also contain a credential that indicates that a persistent file owned by another organisation may be accessed. If the "Permission Request File" is not correctly authenticated the application is not granted any additional permissions but is not prevented from starting for this reason.

The way the user grants rights to the downloaded applications is implementation dependant and is not addressed by this specification.

For DVB-J applications, accessing a resource consists of method calls. Each method call that results in accessing the resource shall throw a security exception as defined in 11.10, "Java permissions" on page 272 and the specifications of the java APIs concerned. For each resource subject to access restriction, the application can test whether it has been granted permissions to access it by using the corresponding java Permission class (See 11.8, "Security" on page 268).

For a DVB-J application to be correctly authenticated, all the class files that the application consists of need to be signed. If, during the execution of the application the MHP detects an unsigned File containing a class or a class file that failed to pass the authentication process (i.e. because its actual hash value does not match the expected hash value), then the class shall be considered as not available.

When an application first requests to retrieve data from a file that is signalled as being signed, but for which the MHP failed to match the computed hash value and the expected hash value, then the API concerned shall fail in a manner consistent with the defined behaviour of that API when the file exists but has no content in it.

NOTE: In order to be efficient, if a directory D contains objects that are likely to frequently change, it is advised to put a signature file in this directory D and to mark the directory D as non authenticated in the hashfile located in the parent directory of D. By doing so, it will limit the propagation of modifications to just one directory.

The authentication of a file is evaluated each time that the file is loaded from a transport connection. File version information in the transport system cannot be assumed to be secure.

Applications authors should be aware that deciding whether to grant a permission or not may, depending on the implementation, involve prompting and asking the end user. The latest point in time when the implementation must decide if an application has a permission or not is when the application either queries the presence of this permission for the first time or when it invokes an action that requires the permission for the first time. Application authors should be aware that in these situations, an implementation may prompt and ask the user. Depending on the implementation, this prompting (if necessary) can also happen at any point in time prior to this (e.g. at the application start up time).

An MHP terminal is required to be able to operate in a mode where it grants permission to provide access to all of the functionality required by the profiles and options that it supports when appropriately requested (e.g. via the permission request file). The mechanism for causing the terminal to operate in this mode is implementation-dependent. The granting of permissions for accessing functionality outside of the claimed MHP profile and options is not required.

NOTE: In the case of permissions represented by a Java class in DVB-J, this means that it will be possible to have such a permission granted if the corresponding class is required in the given profile, and it can be requested in the permission request file.

NOTE: This means that, for example, it must be possible to grant the permission associated with dialling a phone number on a terminal that supports the interactive broadcast profile, even if the terminal implements the interaction channel using a cable modem. In this case, the dialling APIs will fail in a manner consistent with this specification.

## 12.6.2 Permission request file

### 12.6.2.1 File encoding

#### 12.6.2.1.1 XML

The Permission Request File is an XML File. Its syntax is defined by the following DTD. The Name used in the document type declaration shall be "permissionrequestfile".

#### 12.6.2.1.2 MHP 1.0

The `PublicLiteral` to be used for specifying this DTD in document type declarations of the XML files is:

```
"-//DVB//DTD Permission Request File 1.0//EN"
```

and the URL for the `SystemLiteral` is:

```
"http://www.dvb.org/mhp/dtd/permissionrequestfile-1-0.dtd".
```

The DTD is:

```
<!ELEMENT permissionrequestfile
  (file?, capermission?, applifecyclecontrol?, returnchannel?, tuning?,
  servicesel?, userpreferences?, network?, dripfeed?, persistentfilecredential?)+>
<!ATTLIST permissionrequestfile
  orgid CDATA #REQUIRED
  appid CDATA #REQUIRED
>

<!ELEMENT file EMPTY>
<!ATTLIST file
  value          (true|false) "true"
>

<!ELEMENT capermission (casystemid)+>
<!ELEMENT casystemid EMPTY>
```

```

<!ATTLIST casystemid
  entitlementquery    (true|false) "false"
  id                  CDATA #REQUIRED
  mmi                 (true|false) "false"
  messagepassing     (true|false) "false"
  buy                 (true|false) "false"
>

<!ELEMENT applifecyclecontrol EMPTY>
<!ATTLIST applifecyclecontrol
  value              (true|false) "true"
>

<!ELEMENT returnchannel (defaultisp?,phonenumber*)>
<!ELEMENT defaultisp EMPTY>
<!ELEMENT phonenumber (#PCDATA)>

<!ELEMENT tuning EMPTY>
<!ATTLIST tuning
  value              (true|false) "true"
>

<!ELEMENT servicesel EMPTY>
<!ATTLIST servicesel
  value              (true|false) "true"
>

<!ELEMENT userpreferences EMPTY>
<!ATTLIST userpreferences
  write              (true|false) "false"
  read               (true|false) "true"
>

<!ELEMENT network (host)+>
<!ELEMENT host (#PCDATA)>
<!ATTLIST host
  action             CDATA #REQUIRED
>

<!ELEMENT dripfeed EMPTY>
<!ATTLIST dripfeed
  value              (true|false) "true"
>

<!ELEMENT persistentfilecredential (grantoridentifier,expirationdate,filename+,signature,
certchainfileid)>
<!ELEMENT grantoridentifier EMPTY>
<!ATTLIST grantoridentifier
  id                 CDATA #REQUIRED
>
<!ELEMENT expirationdate EMPTY>
<!ATTLIST expirationdate
  date               CDATA #REQUIRED
>
<!ELEMENT filename (#PCDATA)>
<!ATTLIST filename
  write              (true|false) "true"
  read               (true|false) "true"
>
<!ELEMENT signature (#PCDATA)>
<!ELEMENT certchainfileid (#PCDATA)>

```

### 12.6.2.1.3 MHP 1.1

The `PublicLiteral` to be used for specifying this DTD in document type declarations of the XML files is:

```
"-//DVB//DTD Permission Request File 1.1//EN"
```

and the URL for the `SystemLiteral` is:

"http://www.dvb.org/mhp/dtd/permissionrequestfile-1-1.dtd".

The DTD is:

```

<!ELEMENT permissionrequestfile
  (file?, capermission?, applifecyclecontrol?, returnchannel?, tuning?,
  servicesel?, userpreferences?, network?, dripfeed?, persistentfilecredential?,
  applicationstorage?, smartcardaccess?)>

<!ATTLIST permissionrequestfile
  orgid CDATA #REQUIRED
  appid CDATA #REQUIRED
  >

<!ELEMENT file EMPTY>
<!ATTLIST file
  value          (true|false) "true"
  >

<!ELEMENT capermission (casystemid)+>
<!ELEMENT casystemid EMPTY>
<!ATTLIST casystemid
  entitlementquery (true|false) "false"
  id              CDATA #REQUIRED
  mmi            (true|false) "false"
  messagepassing (true|false) "false"
  buy            (true|false) "false"
  >

<!ELEMENT applifecyclecontrol EMPTY>
<!ATTLIST applifecyclecontrol
  value          (true|false) "true"
  >

<!ELEMENT returnchannel (defaultisp?, phonenumber*)>
<!ELEMENT defaultisp EMPTY>
<!ELEMENT phonenumber (#PCDATA)>

<!ELEMENT tuning EMPTY>
<!ATTLIST tuning
  value          (true|false) "true"
  >

<!ELEMENT servicesel EMPTY>
<!ATTLIST servicesel
  value          (true|false) "true"
  >

<!ELEMENT userpreferences EMPTY>
<!ATTLIST userpreferences
  write          (true|false) "false"
  read           (true|false) "true"
  >

<!ELEMENT network (host)+>
<!ELEMENT host (#PCDATA)>
<!ATTLIST host
  action         CDATA #REQUIRED
  >

<!ELEMENT dripfeed EMPTY>
<!ATTLIST dripfeed
  value          (true|false) "true"
  >

<!ELEMENT persistentfilecredential (grantoridentifier, expirationdate, filename+, signature,
certchainfileid)>
<!ELEMENT grantoridentifier EMPTY>
<!ATTLIST grantoridentifier
  id            CDATA #REQUIRED
  >

```



```

<!ELEMENT expirationdate EMPTY>
<!ATTLIST expirationdate
  date          CDATA #REQUIRED
>
<!ELEMENT filename (#PCDATA)>
<!ATTLIST filename
  write          (true|false) "true"
  read           (true|false) "true"
>
<!ELEMENT signature (#PCDATA)>
<!ELEMENT certchainfileid (#PCDATA)>

<!-- ..... new elements in MHP 1.1 ..... -->
<!ELEMENT applicationstorage EMPTY>
<!ATTLIST applicationstorage
  orgid CDATA #REQUIRED
  store (true|false) "false"
  remove (true|false) "false"
  create (true|false) "false"
  removeservice (true|false) "false"
>

<!ELEMENT smartcardaccess EMPTY>

<!ELEMENT privilegedrce EMPTY>
<!ATTLIST privilegedrce
  value (internal|external) "internal"
>

```

## 12.6.2.1.4 Number representation

### 12.6.2.1.4.1 Hex

Unless stated otherwise each hexadecimal value has the following form "0x" N\*hex where "N" specifies the fixed number of significant digits in the value and hex is specified by the following BNF:

```

hex = digit | "A" | "B" | "C" | "D" | "E" | "F" | "a" | "b" | "c" | "d" | "e" | "f"
digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

```

## 12.6.2.2 File integrity

If the permission file is not parsable as defined by the XML parsing rules (see 14.3, "XML notation" on page 355) it shall be ignored and hence no additional permissions are granted.

## 12.6.2.3 Example

```

<?xml version="1.0"?>
<!DOCTYPE permissionrequestfile PUBLIC "-//DVB//DTD Permission Request File 1.0//EN"
"http://www.dvb.org/mhp/dtd/permissionrequestfile-1-0.dtd">
<permissionrequestfile orgid="0x000023d2" appid="0x0020">

  <file value="true"></file>

  <capermission>
    <casystemid
      id="0x1111" messagepassing="true"
      entitlementquery="true" mmi="false">
    </casystemid>
  </capermission>

  <applifecylecontrol value="true"></applifecylecontrol>

  <returnchannel>
    <defaultisp></defaultisp>
    <phonenum>+3583111111</phonenum>
    <phonenum>+3583111112</phonenum>
    <phonenum></phonenum>
  </returnchannel>

```

```

</returnchannel>

<tuning value="false"></tuning>

<servicesel value="true"></servicesel>

<userpreferences read="true" write="false"></userpreferences>

<network>
  <host action="connect">hostname</host>
</network>

<persistentfilecredential>
  <grantoridentifier id="0x0202030"></grantoridentifier>
  <expirationdate date="24/12/2032"></expirationdate>
  <filename read="true" write="false">5/15/dir1/scores</filename>
  <filename read="true" write="false">5/15/dir1/names</filename>
  <signature>023203293292932932921493143929423943294239432
  </signature>
  <certchainfileid>3</certchainfileid>
</persistentfilecredential>

</permissionrequestfile>

```

### 12.6.2.4 Permission request file name and location

The format for the permission request file name is:

```
"dvb.<application name>.perm"
```

The prefix "dvb" identifies this as a well known file specified by this specification. The portion "application name" carries the file name of the initial file of the application. The initial file depends on the application type as is shown in table 125 for the types defined in this specification.

**Table 125 : Application name for different application types**

Application type		File identified for application name
Value	Meaning	
0x0001	DVB-J	The name <i>initial_class_byte</i> , see table 102 on page 230
0x0002	DVB-HTML	The name <i>initial_path_bytes</i> , see table 104 on page 232

This file shall be located in the same directory as the initial file.

### 12.6.2.5 Permission request file

#### 12.6.2.5.1 Minimum permissions

If the permission file does not contain a valid permissionrequestfile element it shall be ignored and hence no additional permissions are granted.

#### 12.6.2.5.2 Syntax and semantics

```

<!ATTLIST permissionrequestfile
  orgid CDATA #REQUIRED
  appid CDATA #REQUIRED
>

```

**orgid:** This attribute is a hexadecimal string (*hex\_string*) that conveys the organisation id of the associated application. The encoding of this value is specified in 12.6.2.1.4.1, "Hex" on page 301.

**appid:** This attribute is a hexadecimal string (*hex\_string*) that conveys the application id of the associated application. The encoding of this value is specified in 12.6.2.1.4.1, "Hex" on page 301.

## 12.6.2.6 Credentials

A credential contains a resource description and is used to allow the owner of this resource (the grantor) to grant to the permission request file's application to access it. In this specification, the only resource that can be contained in a credential is a file (or a set of files of a directory). This type of credential is named `persistentfilecredential` in the XML DTD. The credential contains an expiration date that allow the grantor to grant access to its resource for a limited duration. The credential is signed by the grantor. The signature checking is done by the implementation by getting the certificate of the grantor. The certificate can be found thanks to the information contained in the `certchainfileid` element.

The certificate that leads to the public key of the signature shall be placed either in the same directory as the permission request file containing the credential or in one of its parent directories.

The `grantoridentifier` in the `persistentfilecredential` shall match the `organization_id` contained in the Subject `organizationName` field of the leaf certificate. Likewise, the `grantoridentifier` shall match the owner of the file to which access is granted by this `persistentfilecredential`. If either of these matches is not successful then the file access shall not be granted.

The persistent file credential is transmitted using the following XML DTD syntax:

```
<!ELEMENT persistentfilecredential (grantoridentifier,expirationdate,filename+,signature,
certchainfileid)>
<!ELEMENT grantoridentifier EMPTY>
<!ATTLIST grantoridentifier
  id CDATA #REQUIRED
>
<!ELEMENT expirationdate EMPTY>
<!ATTLIST expirationdate
  date CDATA #REQUIRED
>
<!ELEMENT filename (#PCDATA)>
<!ATTLIST filename
  write (true|false) "true"
  read (true|false) "true"
>
<!ELEMENT signature (#PCDATA)>
<!ELEMENT certchainfileid (#PCDATA)>
```

The following table provide more information about the different elements:

**Table 126 :**

Elements	Comments
<code>grantoridentifier</code>	This element contains the 32 bit organization id identifying the grantor organization. The CDATA attribute <code>id</code> of this element shall have the following BNF syntax: <code>"0x" 0*8hex ; e.g. 0x34243</code> See 12.6.2.1.4.1, "Hex" on page 301.
<code>expirationdate</code>	This element contains the expiration date of this credential. The implementation should ignore the certificate if the date has expired. The CDATA attribute <code>date</code> shall follow the following BNF syntax: <code>2dec "/" 2dec "/" 4dec ; e.g. "01/12/2001"</code> Where the first 2 digits are the day number in the month, the second two digits are the month number in the year and the last 4 digits are the year. All values are in accordance with the Gregorian calendar.

Table 126 :

Elements	Comments
filename	<p>This element contains the filename path and the read/write access rights that are granted on the file. The element consists of a CDATA string following the following BNF syntax:</p> <pre>1*(alphanum   "/"   "*"   "."   "-"   "_"   "?")</pre> <p>"*" can be used as a wildcard to replace any number of character in a file or directory name.  "?" can be used to represent one wildcarded character  "/" is the file separator  "-" used at the end of a pathname indicates (recursively) all files and subdirectories contained in that directory</p> <p>The file path shall not start with a "/". It is relative to the path obtained from the <code>dvb.persistent.root</code> property.  The element has two attributes <code>read</code> and <code>write</code> that can take the value "true" or "false".</p>
signature	<p>This element contains a signature from the grantor. Signature structure is as defined section 12.4.2, "SignatureFile" on page 291. The signature follows the following BNF syntax:</p> <pre>1*dec</pre>
certchainfileid	<p>This element contains the identifier of the leaf certificate, i.e. the "x" in the file name "dvb.certificate.x".</p>

The signature is computed on the binary concatenation of the following fields in the following order with identically the same encoding as in the permission request file:

**Table 127 :**

Fields	Binary content
grantee_identifier.organization_id	32 bits
grantee_identifier.application_id	16 bits
grantor_identifier (organization_id)	32 bits
expiration_date	10 characters (e.g. "10/12/2001") in ASCII
filenames & actions (in the order they appear in the XML document) i.e.:	
<pre> for (i=0;i&lt;filenumber;i++) {   read   write   filepath } </pre>	<pre> 4 or 5 char ("true" or "false") 4 or 5 char ("true" or "false") string in ASCII (without any string termination character) </pre>

The implementation will check the validity of the credential by checking the signature with the grantor's public key that can be found in the grantor's certificate. Certificates are carried by the grantee in the file format defined section 12.4.3, "CertificateFile" on page 292. Certification chain authenticates grantor's certificate. This chain shall derive from one of the root authorities embedded in the MHP.

## 12.6.2.7 File Access

### 12.6.2.7.1 Unsigned applications

Have no access to the persistent storage

### 12.6.2.7.2 Default policy for signed applications

No access to the persistent storage, unless otherwise indicated in the Permission File.

When an application is granted access to the File system, the file access policy is derived from the policy used in the Unix world:

An application owns the files it has created. The root directory of the persistent file namespace is defined by the 'dvb.persistent.root' property as described in 11.5.6, "Persistent Storage API" on page 258. The files owned by an application shall be located in sub-directories below this directory, specifically one of the following two choices:

- a) the sub-directory whose name is the [organisation\\_id](#) of the application concerned.
- b) a sub-directory of the immediately previously defined directory whose name is the [application\\_id](#) of the application concerned.

The encodings of the organisation identifier and application identifier are as defined in 14.5, "Text encoding of application identifiers" on page 357. By default, applications cannot access any file outside the second of the two directories defined immediately above.

An Application can modify the access rights to a file it owns as follows:

- it can grant a read-only access, a write-only access or a read-write access to all applications having the same organisationId value.
- it can grant a read-only access, a write-only access or a read-write access to all applications.
- write access to a directory is required to add or remove an entry in a directory.
- read access to a directory is required to list the contents of a directory or access any of the files contained within in it.
- to read the contents of a file permission to read that file and all directories on the path to the root of the persistent file system is required

An application shall be granted access to a file if it qualifies for such access by application, organisation or world access permissions.

An application's right to access a file is the union of the rights granted by the credential mechanism and the right granted by the file permission mechanism.

See org.dvb.io.persistent see J, "(normative): DVB-J persistent storage API" on page 469.

### 12.6.2.7.3 Permission request syntax

```
<!ELEMENT file EMPTY>
<!ATTLIST file
  value (true|false) "true"
>
```

## 12.6.2.8 CA API

### 12.6.2.8.1 Unsigned applications

An unsigned application cannot access the following methods:

- CAModule.buyEntitlements
- CAModule.openMessageSession
- CAModuleManager.addMMILListener
- CAModule.queryEntitlements
- CAModule.listEntitlements.

### 12.6.2.8.2 Signed applications

By default, an application has limited access to the CA API functions (same default rights as an unsigned application)

In particular, the following method calls are not accessible to an unsigned application:

- CAModule.buyEntitlements
- CAModule.openMessageSession
- CAModuleManager.addMMILListener
- CAModule.queryEntitlements
- CAModule.listEntitlements.

The permission request file requests the MHP to grant additional rights to the application with the ConditionalAccess Permission described below.

### 12.6.2.8.3 Conditional Access Permission syntax

The ConditionalAccess Permission is optional. When not present, the application has the default access rights. When present, the permission request file overrides the default rights for this application.

```
<!ELEMENT capermission (casystemid)+>
<!ELEMENT casystemid EMPTY>
<!ATTLIST casystemid
  entitlementquery      (true|false) "false"
  id                    CDATA #REQUIRED
  mmi                   (true|false) "false"
  messagepassing       (true|false) "false"
  buy                   (true|false) "false"
>
```

The string specifying the CA system IDs has the following syntax:

```
CAIds          = 1*CASystemId | "[" CASystemId "-" CASystemId "]" | "*"
CASystemId     = "0x" 4*hex
```

See 12.6.2.1.4.1, "Hex" on page 301.

### 12.6.2.9 Application lifecycle control policy

Applications shall not launch broadcast applications that are not signalled in the AIT of the same service.

#### 12.6.2.9.1 Unsigned applications

An unsigned broadcast application can launch any application visible in the listing API that is signalled in the same service.

An unsigned application can control (pause, stop, resume) the lifecycle of an application it has launched.

An unsigned application cannot control the lifecycle of an application it has not launched.

#### 12.6.2.9.2 Default policy for Signed applications

By default, a signed application has the same rights as an unsigned application as concerns the application lifecycle control policy.

These default rights can be overridden by the permission request file as described below.

#### 12.6.2.9.3 Syntax

```
<!ELEMENT applifecyclecontrol EMPTY>
<!ATTLIST applifecyclecontrol
  value (true|false) "true"
>
```

**value:** When the boolean value is set to `true`, this means that the application can control the lifecycle of all the applications signalled in the same service. When set to `false` the policy is as in 12.6.2.9.2, "Default policy for Signed applications" on page 307.

### 12.6.2.10 Return channel access policy

#### 12.6.2.10.1 Unsigned applications

An unsigned application may not use the return channel.

#### 12.6.2.10.2 Signed applications

By default, a signed application may not access the return channel, unless otherwise specified by the permission request file. The syntax of the return channel permission is so that it describes the phone numbers that the application may try to dial.

### 12.6.2.10.3 Return channel permission syntax

```
<!ELEMENT returnchannel (defaultisp?,phonenum*)>
<!ELEMENT phonenum (#PCDATA)>
<!ELEMENT defaultisp EMPTY>
```

The syntax of the phone number string is as defined below.

```
phonenum      = "+" digit digits | digits
digits        = "" | digit digits
digit         = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

If default ISP is not present, then the net API shall not use the default ISPnumber.

The presence of the `defaultisp` tag indicates that the application is allowed to use the default ISP connection. If this tag is not present, the application is not allowed to use the default ISP connection.

When a phone number is given, this number defines a prefix of the allowed phone numbers and applications are allowed to call to all numbers that start with one of the prefixes defined in the permission request file, subject to the MHP terminal granting this right.

NOTE: By defining an empty phone number tag (i.e. empty string as the prefix), the application could try to call to any phone number.

### 12.6.2.11 Tuning access policy

#### 12.6.2.11.1 Unsigned applications

An unsigned application may not tune using the Tuning API.

#### 12.6.2.11.2 Signed applications

By default, a signed application may not tune using the Tuning API. However, the right to tune can be requested with the Tuning permission that can be put in the permission request file.

#### 12.6.2.11.3 Tuner Permission syntax

The syntax of this permission is as follows:

```
<!ELEMENT tuning EMPTY><
<!ATTLIST tuning
  value (true|false) "true"
>
```

The value true requests the permission to tune using the Tuning API.

### 12.6.2.12 Service selection policy

#### 12.6.2.12.1 Unsigned applications

Unsigned applications may not select a new service.

#### 12.6.2.12.2 Signed applications

By default, signed applications can select any new service, unless otherwise specified in the permission request file.

Service selection might require return channel access. If an application attempts to do a service selection using a locator that refers to an AIT file delivered over the return channel, and a return channel has not already been established or the application does not have permission to establish a connection via the default isp (see 12.6.2.10, "Return channel access policy" on page 307), then the service selection shall fail in a manner consistent with the defined failure mode for service selection when a security violation is encountered (e.g. a DVB-J application using the service selection API shall be thrown a SecurityException).

Note: See also 12.14.1, "Permission for application loading from the return channel" on page 331



### 12.6.2.12.3 Service Selection Permission

The syntax of the service selection permission is as follows:

```
<!ELEMENT servicesel EMPTY>

<!ATTLIST servicesel
  value (true|false) "true"
>
```

If no service selection permission is present in the permission request file, the application has the default right, i.e. it can select any service.

The value "false" in this item in the permission request file, denies the right to select a new service.

### 12.6.2.13 Media API access policy

The media API is inside the sandbox.

### 12.6.2.14 Inter-application communication policy

#### 12.6.2.14.1 Unsigned applications

Unsigned applications are allowed to communicate with each other through the inter-application communication API. However, an unsigned application is not allowed to communicate with a signed application through the inter-application communication API.

#### 12.6.2.14.2 Signed applications

Signed applications signalled in the same service are allowed to communicate with each other through the inter-application communication API.

No special Permission need be defined.

### 12.6.2.15 User Setting and Preferences access policy

#### 12.6.2.15.1 Unsigned applications

Read access to:

- User Language
- Parental Rating
- DefaultFontSize
- Country Code

An unsigned application cannot access (neither read nor write) to other preferences

#### 12.6.2.15.2 Signed applications

By default, same as unsigned applications. The permission request file may include items that request read access to all user preferences and/or write access to all user preferences.

#### 12.6.2.15.3 Permission syntax

```
<!ELEMENT userpreferences EMPTY>

<!ATTLIST userpreferences
  write (true|false) "false"
  read (true|false) "true"
>
```

True value in the write and read attribute means that the permission for writing and reading, respectively, is requested.

## 12.6.2.16 Network permissions

### 12.6.2.16.1 Unsigned applications

Unsigned applications can not have access to the return channel and therefore do can not access remote network hosts,

### 12.6.2.16.2 Signed applications

For signed applications, the permission request file can contain a set of permissions that specify the hosts and actions for which permissions are requested.

### 12.6.2.16.3 Permission syntax

```
<!ELEMENT network (host)+>
<!ELEMENT host (#PCDATA)>
<!ATTLIST host
  action CDATA #REQUIRED
>
```

The two strings that specific the host and the action shall be formatted as specified in the `java.net.SocketPermission` class as defined in 11.8, "Security" on page 268.

## 12.6.2.17 Dripfeed permissions

### 12.6.2.17.1 Unsigned applications

Has no access to drip feed mode.

### 12.6.2.17.2 Default policy for signed applications

No access to drip feed mode unless otherwise indicated in the Permission file. When an application is granted access to the drip feed mode, it is able to instantiate a `DripFeedDataSource`.

### 12.6.2.17.3 Permission request syntax

```
<!ELEMENT dripfeed EMPTY>
<!ATTLIST dripfeed
  value (true|false) "true"
>
```

## 12.6.2.18 Privileged Runtime Code Extension Permission

**Note:** DVB-J prohibits runtime code extension from data sources less secure than the original program code by virtue of the fact that interoperable DVB-J applications cannot directly create a `ClassLoader`, so this permission does not apply to DVB-J. See 11.10.1.4, "java.lang.RuntimePermission" on page 272 and `DVBClassLoader` in annex H, "(normative): DVB-J fundamental classes" on page 449.

**Note:** In ECMAScript, code executed via runtime code extension has the same permission level as the enclosing application. Thus, granting an unsigned application the ability to do runtime code extension with string data does not give the application the ability to execute a string as privileged code.

### 12.6.2.18.1 Unsigned Applications

Unsigned applications shall be permitted to do runtime code extension with string data from any source.

### 12.6.2.18.2 Signed applications with no permission request file

Signed applications with no permission request file shall be permitted to do runtime code extension with string data from any source.

### 12.6.2.18.3 Signed applications with a permission request file

By default, a signed application with a permission request file is prohibited from doing any runtime code extension from a string. The permission request file may include items that request the ability to do runtime code extension with internal strings only, or with both internal and external strings.

#### 12.6.2.18.4 Permission request syntax

```
<!ELEMENT privilegedrce EMPTY>
<!ATTLIST privilegedrce
  value (internal|external) "internal"
>
```

### 12.6.2.19 Application storage

#### 12.6.2.19.1 Unsigned applications

Have no permission to store applications

#### 12.6.2.19.2 Default policy for signed applications

By default signed applications do not have permission to store any applications.

Permission to store applications can be requested using the permission request file.

#### 12.6.2.19.3 Permission request syntax

```
<!ELEMENT applicationstorage EMPTY>
<!ATTLIST applicationstorage
  orgid CDATA #REQUIRED
  store (true|false) "false"
  remove (true|false) "false"
  create (true|false) "false"
  removeservice (true|false) "false"
>
```

**orgid:** The organization\_id of the organization whose application and stored services this permission request concerns. This field is encoded in hexadecimal form as specified in 14.5, "Text encoding of application identifiers" on page 357.

**store:** When set to "true", requests a permission to store applications of the organization identified by the organization\_id. If set to false, this permission is not requested.

**remove:** When set to "true", requests a permission to remove applications of the organization identified by the organization\_id. If set to false, this permission is not requested.

**create:** When set to "true", requests a permission to create a stored application service with the organization identified by the organization\_id. If set to false, this permission is not requested.

**removeservice:** When set to "true", requests a permission to remove a stored application service with the organization identified by the organization\_id. If set to false, this permission is not requested.

### 12.6.2.20 Non-CA smart card access

#### 12.6.2.20.1 Unsigned applications

Have no permission to access smart cards

#### 12.6.2.20.2 Default policy for signed applications

By default signed applications do not have permission to access smart cards.

Permission to access smart cards can be requested using the permission request file.

### 12.6.2.20.3 Permission request syntax

```
<!ELEMENT smartcardaccess EMPTY>
```

When the smartcardaccess element is included in the permission request file, the permission to access smart cards is requested. If the element is not included in the permission request file, this permission is not requested.

## 12.7 Example of creating an application that can be authenticated

### 12.7.1 Scenario Example

This section is informative and gives an example of how a file system carrying two applications can be organised.

In this example, the file system carries single signed application Xlet1.

The main class of the application is the file `root/Xlet1/classes/Xlet1.class`, other files comprising the application are the following classes:

- `root/Xlet1/classes/foo1.class`
- `root/Xlet1/classes/subclasses/sub1.class`
- `root/Xlet1/classes/subclasses/sub2.class`

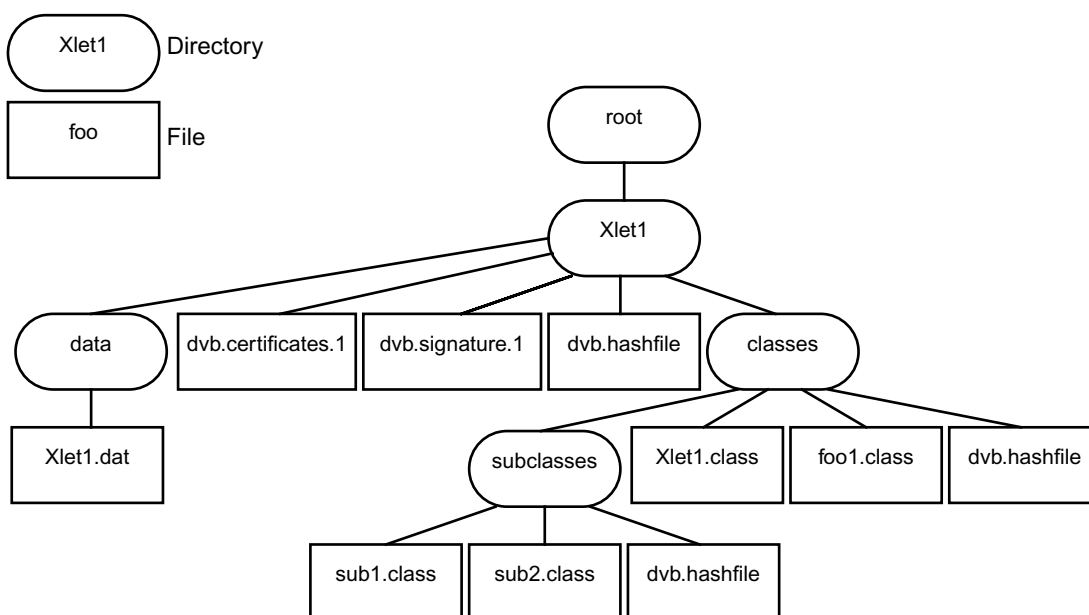
Xlet1 consumes data located in the file `root/Xlet1/data/Xlet1.dat`. This file is not authenticated.

Each subdirectory that contains signed files contains a hashfile.

Assumptions in this example:

- All the \*.class files are signed
- The file `root/Xlet1/data/Xlet1.dat` is not signed
- The only digest algorithm used is MD5

The file structure is shown below:



**Figure 20 : Example of an authenticated file tree**

## 12.7.2 Hashes and signature computations:

### 12.7.2.1 Computation of the hashes of the root/Xlet1/classes/subclasses directory

- a) initialise the MD5 algorithm
- b) apply the contents of the file `root/Xlet1/classes/subclasses/sub1.class` to the MD5 algorithm then apply the contents of the file `root/Xlet1/classes/subclasses/sub2.class` to the MD5 algorithm. The cumulative result from these two files is the result H0
- c) construct the contents of the hash file for the directory `root/Xlet1/classes/subclasses/` as follows:

**Table 128 : root/Xlet1/classes/subclasses/dvb.hashfile**

Field	Comment
1	One digest
1	Type of digest algorithm = MD5
2	Number of entries over which a MD5 hash has been computed
sub1.class	List of names of entries
sub2.class	
H0	MD5 hash of files <code>sub1.class</code> and <code>sub2.class</code>

NOTE: In this example we have chosen that the digests for this directory result from processing the concatenation of the contents of the files in the directory.

- d) create the hash file `root/Xlet1/classes/subclasses/dvb.hashfile` with the above contents

### 12.7.2.2 Computation of the hashes of the of root/Xlet1/classes directory

- e) initialise the MD5 algorithm, compute the MD5 hash H2 using the contents of the file `root/Xlet1/classes/subclasses/dvb.hashfile`
- f) initialise the MD5 algorithm, compute the MD5 hash H3 using the contents of the file `root/Xlet1/classes/Xlet1.class`
- g) initialise the MD5 algorithm, compute the MD5 hash H4 using the contents of the file `root/Xlet1/classes/fool.class`
- h) construct the contents of the hash file for the directory `root/Xlet1/classes/` as follows:

**Table 129 : root/Xlet1/classes/dvb.hashfile**

Field	Comment
3	Three digests
1	Type of digest algorithm = MD5
1	Number of entries over which a MD5 hash has been computed
subclasses	List of names of entries
H2	MD5 hash of <code>subclasses</code> directory
1	Type of digest algorithm = MD5
1	Number of entries over which a MD5 hash has been computed
Xlet1.class	List of names of entries
H3	MD5 hash of file <code>Xlet1.class</code>
1	Type of digest algorithm = MD5
1	Number of entries over which a MD5 hash has been computed
fool.class	List of names of entries
H4	MD5 hash of file <code>fool.class</code>

NOTE: In this example we have chosen to individually hash each object in this directory.

- i) create the hash file `root/Xlet1/classes/dvb.hashfile` with the above contents

### 12.7.2.3 Computation of the hashes of the of root/Xlet1 directory

- j) initialise the MD5 algorithm, compute the MD5 hash H5 using the contents of the file `root/Xlet1/classes/dvb.hashfile`
- k) construct the contents of the hash file for the directory `root/Xlet1/classes/` as follows:

**Table 130 : root/Xlet1/dvb.hashfile**

Field	Comment
2	Two digests
1	Type of digest algorithm = MD5
1	Number of entries over which a MD5 hash has been computed
classes	List of names of entries
H5	MD5 hash of the <code>classes</code> directory
0	Type of digest algorithm = non authenticated data
1	Number of entries over which a MD5 hash has been computed
data	List of names of entries
<no digest in this case>	

NOTE: the `root/Xlet1/classes` entry is the only authenticated entry of the `root/Xlet1` directory

- l) create the hash file `root/Xlet1/dvb.hashfile` with the above contents

### 12.7.2.4 Computation of the signature

- m) initialise the MD5 algorithm, compute the MD5 hash H6 using the contents of the file `root/Xlet1/dvb.hashfile`
- n) RSA-encrypt H6 with the private key corresponding to the public key that can be found in the leaf certificate in the file `root/Xlet1/dvb.certificates.1`. In this example this has serial number 0123456.
- o) ASN.1 encode the following structure:
- AuthorityCertIssuerName: Name of the CA
  - AuthorityCertSerialNumber: 0123456
  - HashSignatureAlgorithm: MD5
  - SignatureValue: result of step (n).
- p) put this structure into the signature file `root/Xlet1/dvb.signature.1`

NOTE: The file system could contain other Xlets in their own authenticated sub-trees. If these use the same certificate chain as Xlet1 (above) then logically the certificates file is replicated at the root of each of the authenticated sub-trees. Some file systems support symbolic links. These may in some cases improve the efficiency of the broadcast.

## 12.8 Procedures for application certificates and signatures

See 12.12, "MHP certification procedures" on page 330.

## 12.9 Certificate management

### 12.9.1 Certificate Revocation Lists

#### 12.9.1.1 Introduction (informative)

Certificates may be revoked prior to their expiration time, e.g. if the broadcaster's private key is assumed to be compromised, or the broadcaster is no longer to be certified by the CA. Each CA publishes a list of revoked certificates, called a CRL (Certificate Revocation List). This contains the list of the serial number of revoked certificates.

During the validation process of a certificate chain, the CRL of each certification authority on the certification path is checked.

#### 12.9.1.2 Distribution of CRLs (informative)

Two routes from the broadcaster to the MHP terminal can be envisaged:

- via the return channel
- in the broadcast MPEG stream

##### 12.9.1.2.1 Distribution via return channel

In the certificate extension fields, there is an optional field called "[CRL Distribution points](#)". This field can hold a URL pointing to a crlFile which can be downloaded.

This approach is suitable for obtaining CRLs relating to TLS authentication of return channel exchanges. It is not suitable for delivering CRLs for broadcast application authentication as:

- not all MHP profiles require return channel support
- MHP terminals may be able to receive broadcast applications when not connected to a return channel
- it would not be acceptable to require a return channel session to authenticate each broadcast application

##### 12.9.1.2.2 Distribution via MPEG stream

For an MHP terminal without a working return channel, the only way to deliver CRLs is via the broadcast MPEG Transport Stream.

### 12.9.1.3 CRL retention

#### 12.9.1.3.1 Requirement

MHP terminals shall retain CRLs in persistent storage because:

- this enhances security as it defends against attacks with signals that filter out the CRL and use a revoked certificate
- it is more efficient to cache CRLs rather than downloading the CRLs for authentication of each application

#### 12.9.1.3.2 Storage requirement

The minimum amount of persistent memory required to store CRLs is addressed in [12.12, "MHP certification procedures"](#) on page 330.

#### 12.9.1.3.3 Storage management

The broadcast CRL and RCMM signalling (see [12.9.2, "Root certificate management"](#) on page 318) manages the use of the persistent storage.

If the CRL of a non-root certificate CA become's too large the CA's certificate itself can be revoked by its parent CA who adds it to their CRL. For root certificates, the RCMM mechanism can be used.

The broadcaster policy in this area is addressed in [12.12, "MHP certification procedures"](#) on page 330.

#### 12.9.1.4 CRL file location and naming convention

For CRLs that are authenticated by a broadcast certificate the format of the name of files carrying CRLs shall be:

```
"dvb.crl.x"
```

In this case the "x" portion of the filename corresponds to the "x" portion of a certificate filename for the certificate file that authenticates the CRL.

For CRLs that are authenticated by a root certificate the format of the name of files carrying CRLs shall be:

```
"dvb.crl.root.x"
```

In this case the "x" portion of the filename is just a discriminator to ensure non-collision of CRL file names in the event that there is more than one in this directory.

The CRL filename may not be constant through time or across broadcasts. So, implementations shall not rely on this filename when caching the CRL.

All CRL files shall be located in a subdirectory of the root of the file system called "dvb.crl". The location of certificate files authenticating the CRL files shall follow the same rules as for the location of certificates relative to a signature file. That is the certificate files shall either be in the dvb.crl directory or in the root directory. See [12.4.3.5, "CertificateFile location and naming conventions"](#) on page 293.

#### 12.9.1.5 Operational model

Receivers are expected to inspect the set of CRL files periodically and cache the revocation information for future use. This inspection process can assume that a file system will not normally carry certificates revoked by CRL files carried in the same file system. So, inspection of the CRL set in a file system is not a precondition to authenticating other files in that file system.

This specification does not address the reliability with which implementations are required to consider certificate revocations when authenticating files.

#### 12.9.1.6 Examples

##### 12.9.1.6.1 Revocation of a broadcaster's certificate

If the broadcaster B's certificate is compromised:

- a) the certification authority CA01 adds the serial number of broadcaster B's certificate to its CRL
- b) CA01 then sends the new CRL file to the broadcasters that use CA01
- c) these broadcasters (broadcaster A for instance) broadcast the new CRL file

As soon as an MHP terminal has downloaded the new CRL file (after selecting one of broadcaster A's channel), the MHP terminal's CRL cache in persistent storage is updated. The MHP terminal is then protected against any malicious usage of the compromised certificate.

To continue authenticating applications broadcast "B" will require a new certificate.

##### 12.9.1.6.2 Revocation of a CA's certificate.

If the CRL of CA01 becomes too big, CA01's certificate could be revoked:

- a) the root certification authority RCA0 adds the serial number of CA01's certificate to its CRL
- b) RCA0 sends the new CRL file to the broadcasters that use RCA0
- c) these broadcasters broadcast the new CRL file

As soon as an MHP terminal has downloaded the new CRL file, its CRL cache in persistent storage for RCA0 is updated and CA01's CRL is removed from the cache (as CA01 has been revoked).



### 12.9.1.7 CRL format

Each CRL file contains the CRL of one certification authority.

The encoding of the CRL is defined in the ITU-T X.509 [54] is reproduced below for information. The fields Version, Time, CertificateSerialNumber correspond to fields with the same names in the certificate see 12.11, "The internet profile of X.509 (informative)" on page 322:

```
CertificateList ::= SEQUENCE {
    tbsCertList          TBSCertList,
    signatureAlgorithm   AlgorithmIdentifier,
    signatureValue BIT STRING }

TBSCertList ::= SEQUENCE {
    version              Version OPTIONAL,
                        -- if present, shall be v2
    signature            AlgorithmIdentifier,
    issuer               Name,
    thisUpdate           Time,
    nextUpdate           Time OPTIONAL,
    revokedCertificates  SEQUENCE OF SEQUENCE {
    userCertificate      CertificateSerialNumber,
    revocationDate      Time,
    crlEntryExtensions  Extensions OPTIONAL
                        -- if present, shall be v2
    } OPTIONAL,
    crlExtensions        [0] EXPLICIT Extensions OPTIONAL
                        -- if present, shall be v2
}
```

### 12.9.1.8 Profile of CRL

The profile of fields that correspond to fields in the certificate follow the profile in 12.5, "Profile of X.509 certificates for authentication of applications" on page 294. This applies to the following fields:

**signatureAlgorithm** : follows "signatureAlgorithm" on page 294.

**signatureValue**: follows "signatureValue" on page 294.

**version**: follows "version" on page 294.

**signature**: follows "signatureAlgorithm" on page 294.

**issuer**: follows "issuer" on page 295.

**thisUpdate**: Publication date of this CRL. Follows the encoding of Time used for validity. See "validity" on page 295.

**nextUpdate**: Publication date of the next version of the CRL. Follows the encoding of Time used for validity. See "validity" on page 295.

**userCertificate**: SerialNumber of the revoked certificate. It is used to identify the revoked certificate.

The serial number shall be unique for a given CA. So, the pair [issuerName, serialNumber] shall be unique.

**revocationDate**: Date of revocation for a given certificate. Follows the encoding of Time used for validity. See "validity" on page 295.

**crlExtensions**: The syntax of the Extensions element is reproduced on "Extensions" on page 327. This element allows multiple extension elements to be carried. The set of extensions defined for certificates and CRLs is reproduced at 12.11.2, "Standard certificate extensions" on page 327.

The following crlExtension shall be supported:

- AuthorityKeyIdentifier as defined in section 12.4.2.1 on page 291.

This extension identifies the certificate that is needed to check the signature of the CRL.

Other crlExtensions are optional.

### 12.9.1.9 CRL Processing

List of revoked serial numbers shall be kept in persistent storage in set top box.

When an MHP terminal finds a file with the file name format given in [12.4.3.5, "CertificateFile location and naming conventions" on page 293](#), it shall do the following sequence:

- a) Get the field thisUpdate and compare it with the last update for the CRL. If thisUpdate is not after the last update, ignore the CRL.
- b) Check the signature of the CRL. If it is invalid, ignore the message.
- c) Process the content of the CRL message:
  - store the list of serial numbers of revoked certificates in persistent storage.
  - update the stored value of thisUpdate with the value from the CRL.
- d) If a CA's certificate has been revoked, remove its CRL if it was stored in the MHP terminal.

During the validation process of a certificate chain, the CRL of each certification authority on the certification path is checked.

## 12.9.2 Root certificate management

### 12.9.2.1 Introduction

Every compliant MHP terminal will have to maintain a set of X.509 root certificates in persistent storage. These root certificates will be placed in the MHP terminal by its manufacturer during the manufacturing process.

It could be necessary to update the set of root certificates for MHP terminals that are already deployed. Possible reasons that could require such an update include:

- a root certificate becoming compromised
- technical developments (such as the emergence of factorisation algorithms) that require the use of greater key lengths in root certificates to provide adequate security
- retirement of a certificate due to its age

So, It is necessary to have a standard mechanism to update this set.

NOTE: A manufacturer specific mechanism could require a different message for each manufacturer and so would be more expensive in terms of broadcast bandwidth.

The mechanism specified here uses messages called RCMM (Root Certificate Management Messages). These messages contain a set of new root certificates to add and a reference to the root certificates to remove.

### 12.9.2.2 Security of RCMM

RCMM are authenticated by multiple signatures. An RCMM message will be accepted by an MHP terminal if and only if it has at least N signatures.

The initial value of N and the maximum value of N that MHP terminals will ever be asked to support are specified in [12.12, "MHP certification procedures" on page 330](#).

NOTE: The initial value of N is expected to be 2

The use of multiple signatures guarantees that the set of root certificates can be updated securely even if one of the root certificates has been compromised.

RCMM can update the number of signatures required for future RCMM using the nextNbOfSignature field.

The RCMM message shall be signed with the key of the certificates to be removed.

### 12.9.2.3 Format of RCMM

The encoding of RCMM is ASN.1 DER (see ASN.1 [57]):

```

URCMM ::= SEQUENCE {
    issuer                Name,
    thisUpdate            Time,
    nextNbOfSignatures   INTEGER OPTIONAL,
    addedCertificates     SET OF Certificate
    removedCertificates   SET OF CertificatesReference }

CertificatesReference ::= SEQUENCE {
    issuerName            Name,
    serialNumber          CertificateSerialNumber }

```

**issuer:** Identification of the certification authority which has issued the message.

**thisUpdate:** Date of the issue of the message.

**nextNbOfSignatures:** This field could be used to change the minimum number of valid signatures required for an RCMM message. This value will be applied to the next RCMMs not to itself!

**addedCertificates:** List of root certificates to be added in persistent storage

**removedCertificates:** Reference of the root certificates to be removed from persistent storage

```

RCMM ::= SEQUENCE {
    uRcmm                URCMM,
    signatures            SET OF SignatureInfo }

SignatureInfo ::= SEQUENCE {
    signerName            Name,
    signatureAlgorithm    AlgorithmIdentifier,
    signatureValue        BIT STRING }

```

NB: The signatures are computed on the whole content of uRCMM.

**issuerName:** Name of the issuer of the root certificate to remove (for a self signed root certificate this field is equal to the subject name).

**serialNumber:** Serial number of the root certificate to remove.

### 12.9.2.4 Distribution of RCMM

RCMM are distributed to the MHP terminals in the broadcast MPEG Transport Stream. The RCMM from a particular CA are supplied to at least the broadcasters that use that CA.

The RCMM shall be placed in a file named:

```
"dwb.rcmm"
```

The RCMM files are inserted on a sample basis specified by the CA. This file shall be located in root directories of the object carousels broadcast.

### 12.9.2.5 RCMM Processing

When an MHP terminal finds RCMM in a root directory, it shall perform the following sequence of operations:

- a) Get the field `thisUpdate` and compare it with the last update. If `thisUpdate` is not after the last update, ignore the message.
- b) If there is a `nextNbOfSignatures` in the RCMM, and if `nextNbOfSignatures` will be greater than the number of remaining root CAs (i.e. the number of root CA that would remain if the RCMM was processed), ignore the RCMM.

- c) Get the number signatures from the RCMM, if this number is lower than the minimum required, ignore the message.
- d) If the RCMM contains references to root certificates to remove, check this RCMM is signed with the keys belonging to the certificates to be removed. If at least one of these signatures is missing, ignore the RCMM message.
- e) Check all the signatures of the RCMM. If any of the signatures is invalid, ignore the message.
- f) Process the content of the RCMM message, add and remove root certificates according to the RCMM message.
- g) Store in persistent storage the date of thisUpdate as the date of the last update.
- h) If a root certificate is removed, the CRLs associated are also removed.
- i) If there is a `nextNbOfSignatures` in the RCMM, replace the minimum number of signature required by the `nextNbOfSignatures`.
- j) Since this process will permanently modify the terminal behaviour and is highly sensitive for security, this is allowed to have an implementation dependent enabling / confirmation step prior to performing the action (e.g. it may prompt the end user for verification). Such an enabling / confirmation step is allowed to reject this RCMM action.

The implementation shall ensure the following:

- The integrity of the persistent storage shall be kept if the power supply fails for any reason during the processing of the RCMM message.
  - i.e. If the power is switched off during the processing of the RCMM message, the set of root certificates shall remain as if processing of the RCMM message had not started.
- The minimum amount of memory reserved to store the list of root certificate is specified in 12.12, "MHP certification procedures" on page 330. If there is not enough persistent storage available to process an RCMM message, it shall be ignored (to prevent inconsistencies).

### 12.9.2.6 Example: Renewal of a root certificate

Let's assume the root certification authority RCA has two certificates in each MHP terminal: RC0 and RC1. If RC0 is compromised, RCA may wish to renew this certificate using the following steps:

- a) RCA generates a new key pair and a new self signed certificate RC2.
- b) RCA provides new certificates signed by RC1 to all the entities authenticated by RCA.
- c) Wait until all entities authenticated by RCA have switched to the new certificates and have stopped using RC0.
- d) RCA generates an RCMM message to add RC2 and to remove RC0. This RCMM will be double signed by RC0 and RC1 keys.
- e) RCA will deliver this RCMM message to the broadcasters to update the MHP terminals.
  - (The broadcasting period should be long enough to update almost all of the MHP terminals in the field).
- f) RCA will provide RC1 and RC2 to set top box manufacturers as the new list of root certificates to put in set top boxes.

## 12.10 Security on the return channel

General purpose security for the return channel is provided by the TLS (Transport Layer Security) protocol as described in RFC 2246 [63].

### 12.10.1 MHP functionality

When implementing return channel security the MHP shall:

- implement the cipher suites identified in section 12.10.2

The MHP is not required to implement the following:

- the server part of the TLS protocol
- compliance with SSL 3.0
- TLS client authentication

## 12.10.2 TLS cipher suites

The minimum set of cypher tools that implementations of the MHP profile of TLS shall implement are:

- RSA ()
- MD5
- SHA-1
- DES

More detail of this requirement is given in table 131 (see RFC 2246 [63] for definition of the terms) which identifies which methods are required in an MHP.

**Table 131 : Profile of cipher suites that implementations are required to support**

CipherSuite	Key Exchange	Cipher	Hash	Value (hex)	MHP status
TLS_NULL_WITH_NULL_NULL	NULL	NULL	NULL	00, 00	Required
TLS_RSA_WITH_NULL_MD5	RSA	NULL	MD5	00, 01	Required
TLS_RSA_WITH_NULL_SHA	RSA	NULL	SHA-1	00, 02	Required
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA	RSA_EXPORT	DES40_CBC	SHA-1	00, 08	Required
TLS_RSA_WITH_DES_CBC_SHA	RSA	DES_CBC	SHA-1	00, 09	Required
TLS_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES_EDE_CBC	SHA-1	00, 0A	Required
DVB_RSA_WITH_3DES_EDE112_SHA	RSA	3DES_EDE112_CBC	SHA-1	FF, 01	Required

## 12.10.3 The EDE112 cipher

The DVB\_RSA\_WITH\_3DES\_EDE112\_SHA cipher suite is introduced by this specification. It is the following triple encryption using 2 different DES keys:

$$C = E_{K1}(D_{K2}(E_{K1}(P)))$$

$$P = D_{K1}(E_{K2}(D_{K1}(C)))$$

This algorithm is a variant of TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA where the keys that are used in first and third stage of the encryption process are identical (i.e.  $k_1 = k_3$ ).

## 12.10.4 Downloading of certificates for TLS

### 12.10.4.1 Introduction

Before the TLS connection can be established, the MHP has to ensure that the certificate list sent by a server contains at least one trusted certificate. In computer environment, this is simply done by checking the list of certificates against one certificate that is resident in the computer.

In the MHP environment, a downloadable application can establish a TLS session. This can be used for e.g. sensitive transactions. In such a scenario, the application knows which server to connect to, and also knows one certificate against which it can check that a given certificate chain contains the expected certificate that it knows and trusts.

The API that is used by a downloadable application is described in section 11.8.2, "APIs to Support TLS / SSL Over the Return Channel" on page 269. The process of server authentication involves the checking of the certificate chain sent by the TLS server. As described above, the application may trust a certificate that the MHP does not know about. This certificate needs then to be downloaded by the MHP before the TLS session can be established.

This section specifies how the MHP terminal identifies and manages the TLS certificates that are downloaded along with the application and how verification (or otherwise) is presented to the application).

## 12.10.4.2 Usage of certificate in TLS

### 12.10.4.2.1 When certificates are delivered with the application

One or several TLS root certificates can be optionally broadcast along with the application.

When the certificate chain sent by the TLS server is not compatible with any of the TLS root certificates sent with the application an `IOException` will be thrown.

#### 12.10.4.2.1.1 Certificate file naming and location

To facilitate certificate chain checking the name of the certificate file shall be:

```
dvb.tls.organisation_id.application_id.x
```

where:

- "x" is an optional string discriminating certificates where necessary.
- the encoding of `organisation_id` and `application_id` are specified in 14.5, "Text encoding of application identifiers" on page 357.

Location of the TLS certificates is application type dependant. See table 132.

**Table 132 : TLS certificate locator semantics**

application_type	description
0x0000	reserved
0x0001	For DVB-J the TLS certificate(s) are placed in the base directory of the application as defined in 10.9.2, "DVB-J application location descriptor" on page 230.
0x0002	For DVB-HTML the TLS certificate(s) are placed at the physical root of the application as defined in 10.10.2, "DVB-HTML application location descriptor" on page 232.
0x0003...0xFFFF	

#### 12.10.4.2.1.2 Certificate authentication

To be considered valid TLS certificates the certificate files shall be authenticated members of the same authenticated sub-tree as the application.

#### 12.10.4.2.2 When no certificates are provided

When there are no TLS certificates sent with the application then the implementation will allow connection to be established to any server. The application can then use the JSSE API (see 11.8.2 on page 269) to retrieve the certificate chain and check that it contains what the application requires. In such a case both name and public keys need to be checked by the application if the application wants to be sure of the remote server.

## 12.11 The internet profile of X.509 (informative)

The text that follows summarises the technical features of RFC 2459 [58] and references the different profile decisions made for the different MHP application areas.

## 12.11.1 Main part of the certificate

### 12.11.1.1 Certificate

```
Certificate ::= SEQUENCE {
    tbsCertificate      TBSCertificate,
    signatureAlgorithm  AlgorithmIdentifier,
    signatureValue      BIT STRING }
```

### 12.11.1.2 signatureAlgorithm

The signatureAlgorithm field contains the identifier for the cryptographic algorithm used by the CA to sign this certificate.

An algorithm identifier is defined by the following ASN.1 structure:

```
AlgorithmIdentifier ::= SEQUENCE {
    algorithm          OBJECT IDENTIFIER,
    parameters        ANY DEFINED BY algorithm OPTIONAL }
```

The algorithm identifier is used to identify a cryptographic algorithm. The OBJECT IDENTIFIER component identifies the algorithm. The contents of the optional parameters field will vary according to the algorithm identified.

This field MUST contain the same algorithm identifier as the signature field in the sequence TBSCertificate.

See 12.5.1, "signatureAlgorithm" on page 294.

NOTE: RFC 2459 [58] section 7.2 lists the signature algorithms supported by that profile.

For all of the currently specified algorithms possible non-NULL parameters shall be ignored.

### 12.11.1.3 signatureValue

The signatureValue field contains a digital signature computed upon the ASN.1 DER encoded tbsCertificate. The ASN.1 DER encoded tbsCertificate is used as the input to the signature function. This signature value is then ASN.1 encoded as a BIT STRING and included in the Certificate's signature field.

NOTE: RFC 2459 [58] section 7.2 describes in detail this process for the algorithms supported by that profile.

By generating this signature, a CA certifies the validity of the information in the tbsCertificate field. In particular, the CA certifies the binding between the public key material and the subject of the certificate.

See 12.5.2, "signatureValue" on page 294.

### 12.11.1.4 tbsCertificate

The field contains the names of the subject and issuer, a public key associated with the subject, a validity period, and other associated information. The tbscertificate may also include extensions.

The pair issuer / serialNumber uniquely identifies the certificate.

```
TBSCertificate ::= SEQUENCE {
    version          [0] EXPLICIT Version DEFAULT v1,
    serialNumber     CertificateSerialNumber,
    signature        AlgorithmIdentifier,
    issuer           Name,
    validity         Validity,
    subject          Name,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    issuerUniqueID  [1] IMPLICIT UniqueIdentifier OPTIONAL,
    -- If present, version shall be v2 or v3
    subjectUniqueID [2] IMPLICIT UniqueIdentifier OPTIONAL,
    -- If present, version shall be v2 or v3
    extensions      [3] EXPLICIT Extensions OPTIONAL
    -- If present, version shall be v3 }
```

### 12.11.1.5 version

This field describes the version of the encoded certificate. When extensions are used, as expected in this profile, use X.509 version 3 (value is 2). If no extensions are present, but a UniqueIdentifier is present, use version 2 (value is 1). If only basic fields are present, use version 1 (the value is omitted from the certificate as the default value).

Implementations SHOULD be prepared to accept any version certificate. At a minimum, conforming implementations MUST recognize version 3 certificates.

**Version** ::= INTEGER { v1(0), v2(1), v3(2) }

Generation of version 2 certificates is not expected by implementations based on this profile.

See 12.5.3, "version" on page 294.

### 12.11.1.6 serialNumber

The serial number is an integer assigned by the CA to each certificate. It MUST be unique for each certificate issued by a given CA (i.e., the issuer name and serial number identify a unique certificate).

**CertificateSerialNumber** ::= INTEGER

### 12.11.1.7 signature

This field contains the algorithm identifier for the algorithm used by the CA to sign the certificate.

This field MUST contain the same algorithm identifier as the [signatureAlgorithm](#) field in the sequence [Certificate](#). The contents of the optional parameters field will vary according to the algorithm identified.

If the signatureAlgorithm is different from the algorithm identifier in the signature field, the signature shall be rejected as being inconsistent.

See 12.5.1, "signatureAlgorithm" on page 294.

NOTE: RFC 2459 [58] section 7.2 lists the supported signature algorithms for that profile.

### 12.11.1.8 issuer

The issuer field identifies the entity who has signed and issued the certificate. The issuer field MUST contain a non-empty distinguished name (DN). The issuer field is defined as the X.501 type Name. ITU-T X.501 [53] Name is defined by the following ASN.1 structures:

**Name** ::= CHOICE {  
    RDNSequence }

**RDNSequence** ::= SEQUENCE OF [RelativeDistinguishedName](#)

**RelativeDistinguishedName** ::= SET OF [AttributeTypeAndValue](#)

**AttributeTypeAndValue** ::= SEQUENCE {  
    type            [AttributeType](#),  
    value           [AttributeValue](#) }

**AttributeType** ::= OBJECT IDENTIFIER

**AttributeValue** ::= ANY DEFINED BY [AttributeType](#)

**DirectoryString** ::= CHOICE {  
    teletexString                    TeletexString (SIZE (1..MAX)),  
    printableString                  PrintableString (SIZE (1..MAX)),  
    universalString                  UniversalString (SIZE (1..MAX)),  
    utf8String                        UTF8String (SIZE (1..MAX)),  
    bmpString                         BMPString (SIZE (1..MAX)) }



The Name describes a hierarchical name composed of attributes, such as country name, and corresponding values, such as US. The type of the component AttributeValue is determined by the AttributeType; in general it will be a DirectoryString.

See 12.5.4, "issuer" on page 295.

### 12.11.1.9 validity

The certificate validity period is the time interval during which the CA warrants that it will maintain information about the status of the certificate. The field is represented as a SEQUENCE of two dates: the date on which the certificate validity period begins (notBefore) and the date on which the certificate validity period ends (notAfter). Both notBefore and notAfter may be encoded as UTCTime or GeneralizedTime.

CAs conforming to this profile MUST always encode certificate validity dates through the year 2049 as UTCTime; certificate validity dates in 2050 or later MUST be encoded as GeneralizedTime.

```
Validity ::= SEQUENCE {
    notBefore      Time,
    notAfter       Time }

Time ::= CHOICE {
    utcTime        UTCTime,
    generalTime    GeneralizedTime }
```

#### 12.11.1.9.1 UTCTime

The universal time type, UTCTime, is a standard ASN.1 type intended for representation of dates and time. UTCTime specifies the year through the two low order digits and time is specified to the precision of one minute or one second. UTCTime includes either Z (for Zulu, or Greenwich Mean Time) or a time differential.

For the purposes of this profile, UTCTime values MUST be expressed Greenwich Mean Time (Zulu) and MUST include seconds (i.e., times are YYMMDDHHMMSSZ), even where the number of seconds is zero. Conforming systems MUST interpret the year field (YY) as follows:

- Where YY is greater than or equal to 50, the year shall be interpreted as 19YY; and
- Where YY is less than 50, the year shall be interpreted as 20YY.

#### 12.11.1.9.2 GeneralizedTime

The generalized time type, GeneralizedTime, is a standard ASN.1 type for variable precision representation of time. Optionally, the GeneralizedTime field can include a representation of the time differential between local and Greenwich Mean Time.

For the purposes of this profile, GeneralizedTime values MUST be expressed Greenwich Mean Time (Zulu) and MUST include seconds (i.e., times are YYYYMMDDHHMMSSZ), even where the number of seconds is zero. GeneralizedTime values MUST NOT include fractional seconds.

See 12.5.5, "validity" on page 295.

### 12.11.1.10 subject

The subject field identifies the entity associated with the public key stored in the SubjectPublicKeyInfo field. It thus represents the entity whose public key is certified. It is encoded as a Distinguished Name (see "issuer" on page 324). This name must be unique for each subject entity certified by one CA as defined by the issuer field.

#### 12.11.1.10.1 issuerUniqueID

This field is optional and appears in X.509 v2 or v3 only. It enables to reuse the IssuerName over time. It is redundant with the issuer Name, and it is proposed here that this field be not parsed by the client.

### 12.11.1.10.2 subjectUniqueID

This field is optional and appears in X.509 v2 or v3 only. It enables to reuse the subjectName over time. It is redundant with the [subject Name](#), and it is proposed here not to use this field.

The subject name may be carried in the subject field and/or the subjectAltName extension. If the subject is a CA (e.g., the basic constraints extension, as discussed in [RFC 2459 \[58\]](#) section 4.2.1.10, is present and the value of cA is TRUE,) then the subject field MUST be populated with a non-empty distinguished name matching the contents of the issuer field (see [RFC 2459 \[58\]](#) section 4.1.2.4) in all certificates issued by the subject CA. If subject naming information is present only in the subjectAltName extension (e.g., a key bound only to an email address or URI), then the subject name MUST be an empty sequence and the subjectAltName extension MUST be critical.

Where it is non-empty, the subject field MUST contain an X.500 distinguished name (DN). The DN MUST be unique for each subject entity certified by the one CA as defined by the issuer name field. A CA may issue more than one certificate with the same DN to the same subject entity.

The subject name field is defined as the X.501 type Name. Implementation requirements for this field are those defined for the issuer field (see [RFC 2459 \[58\]](#) section 4.1.2.4). When encoding attribute values of type DirectoryString, the encoding rules for the issuer field MUST be implemented. Implementations of this specification MUST be prepared to receive subject names containing the attribute types required for the issuer field. Implementations of this specification SHOULD be prepared to receive subject names containing the recommended attribute types for the issuer field. The syntax and associated object identifiers (OIDs) for these attribute types are provided in the [ASN.1](#) modules in [RFC 2459 \[58\]](#) Appendices A and B. Implementations of this specification MAY use these comparison rules to process unfamiliar attribute types (i.e., for name chaining). This allows implementations to process certificates with unfamiliar attributes in the subject name.

In addition, legacy implementations exist where an RFC 822 name is embedded in the subject distinguished name as anEmailAddress attribute. The attribute value for EmailAddress is of type IA5String to permit inclusion of the character '@', which is not part of the PrintableString character set. EmailAddress attribute values are not case sensitive (e.g., "fanfeedback@redsox.com" is the same as "FANFEEDBACK@REDSOX.COM").

Conforming implementations generating new certificates with electronic mail addresses MUST use the rfc822Name in the subject alternative name field (see [RFC 2459 \[58\]](#) section 4.2.1.7) to describe such identities. Simultaneous inclusion of the EmailAddress attribute in the subject distinguished name to support legacy implementations is deprecated but permitted.

### 12.11.1.11 SubjectPublic Key Info

This field is used to carry the public key which is certified and identifies the algorithm with which the key is used. The algorithm is identified using the [AlgorithmIdentifier](#) structure (see "[signatureAlgorithm](#)" on page 323) and the public key is represented as a bitstring.

```
SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm           AlgorithmIdentifier,
    subjectPublicKey    BIT STRING }
```

See 12.5.7, "[SubjectPublic Key Info](#)" on page 295.

NOTE: [RFC 2459 \[58\]](#) section 7.3 lists the supported algorithms for that profile.

### 12.11.1.12 Unique Identifiers

These fields may only appear if the version is 2 or 3. The subject and issuer unique identifiers are present in the certificate to handle the possibility of reuse of subject and/or issuer names over time. This profile recommends that names not be reused for different entities and that Internet certificates not make use of unique identifiers. CAs conforming to this profile SHOULD NOT generate certificates with unique identifiers. Applications conforming to this profile SHOULD be capable of parsing unique identifiers and making comparisons.

```
UniqueIdentifier ::= BIT STRING
```

See 12.5.8, "[Unique Identifiers](#)" on page 296.

### 12.11.1.13 Extensions

This field may only appear if the version is 3 and is optional. If present, this field is a SEQUENCE of one or more certificate extensions.

```
Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension
```

```
Extension ::= SEQUENCE {
    extnID      OBJECT IDENTIFIER,
    critical    BOOLEAN DEFAULT FALSE,
    extnValue   OCTET STRING }
```

The extensions are defined in ITU-T X.509 [54].

See 12.5.9, "Extensions" on page 296.

## 12.11.2 Standard certificate extensions

See table 123, "Profile for standard certificate extensions" on page 296.

### 12.11.2.1 Authority key identifier

This extension is used where an issuer has multiple signing keys. It provides a means of finding the public key that can be used to check the signature of the certificate.

The identification can be based on either the keyIdentifier or the on the pair (authorityCertIssuer, AuthorityCertSerialNumber). It is recommended to use the pair (authorityCertIssuer, AuthorityCertSerialNumber) instead of the keyIdentifier because there is no common agreement on the way to compute a unique KeyIdentifier.

```
AuthorityKeyIdentifier ::= SEQUENCE {
    keyIdentifier      [0] KeyIdentifier      OPTIONAL,
    authorityCertIssuer [1] GeneralNames      OPTIONAL,
    authorityCertSerialNumber [2] CertificateSerialNumber OPTIONAL }
```

### 12.11.2.2 Subject key identifier

This extension provides of identifying certificates that contain a particular public key.

```
SubjectKeyIdentifier ::= KeyIdentifier
```

### 12.11.2.3 Key usage

The key usage extension defines the purpose of the key contained in the certificate.

```
keyUsage ::= BIT STRING {
    digitalSignature      (0),
    nonRepudiation        (1),
    keyEncipherment       (2),
    dataEncipherment      (3),
    keyAgreement          (4),
    keyCertSign           (5),
    cRLSign               (6),
    encipherOnly          (7),
    decipherOnly          (8)
}
```

### 12.11.2.4 Private key usage period

This field is used to defined a period of validity for the private key which is different from the period of validity of the certificate. This is only meaningful for digital signature keys.

```
privateKeyUsagePeriod EXTENSION ::= {
    SYNTAX      PrivateKeyUsagePeriod
    IDENTIFIED BY id-ce-privateKeyUsagePeriod }
```

```
PrivateKeyUsagePeriod ::= SEQUENCE {
    notBefore      [0] GeneralizedTime OPTIONAL,
```

```

notAfter          [1] GeneralizedTime OPTIONAL }
( WITH COMPONENTS {..., notBefore PRESENT} |
  WITH COMPONENTS {..., notAfter PRESENT} )

```

### 12.11.2.5 Certificate policies

This field is used to define a policy defining the purpose for which the certificate may be used.

Applications may have a list of specific policies they will accept, the certificate validation software must be able to compare the policy OIDs found in the certificate to that list.

```

certificatePolicies EXTENSION ::= {
  SYNTAX      CertificatePoliciesSyntax
  IDENTIFIED BY id-ce-certificatePolicies }

```

```

CertificatePoliciesSyntax ::= SEQUENCE SIZE (1..MAX) OF PolicyInformation

```

```

PolicyInformation ::= SEQUENCE {
  policyIdentifier      CertPolicyId,
  policyQualifiers     SEQUENCE SIZE (1..MAX) OF
                      PolicyQualifierInfo OPTIONAL }

```

```

CertPolicyId ::= OBJECT IDENTIFIER

```

```

PolicyQualifierInfo ::= SEQUENCE {
  policyQualifierId      CERT-POLICY-QUALIFIER.&id
                        ({{SupportedPolicyQualifiers}}),
  qualifier              CERT-POLICY-QUALIFIER.&Qualifier
                        ({{SupportedPolicyQualifiers}}{@policyQualifierId})
                        OPTIONAL }

```

```

SupportedPolicyQualifiers CERT-POLICY-QUALIFIER ::= { ... }

```

### 12.11.2.6 Policy mappings

This field is used to define equivalence between policies from different CA's policy Domain.

```

policyMappings EXTENSION ::= {
  SYNTAX      PolicyMappingsSyntax
  IDENTIFIED BY id-ce-policyMappings }

```

```

PolicyMappingsSyntax ::= SEQUENCE SIZE (1..MAX) OF SEQUENCE {
  issuerDomainPolicy      CertPolicyId,
  subjectDomainPolicy    CertPolicyId }

```

### 12.11.2.7 Subject Alternative Name

This field is used to define additional identities for the subject name of the certificate (such as an internet email address).

```

subjectAltName EXTENSION ::= {
  SYNTAX      GeneralNames
  IDENTIFIED BY id-ce-subjectAltName }

```

```

GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName

```

```

GeneralName ::= CHOICE {
  otherName          [0] INSTANCE OF OTHER-NAME,
  rfc822Name         [1] IA5String,
  dNSName            [2] IA5String,
  x400Address        [3] ORAddress,
  directoryName      [4] Name,
  ediPartyName       [5] EDIPartyName,
  uniformResourceIdentifier [6] IA5String,
  iPAddress          [7] OCTET STRING,
  registeredID       [8] OBJECT IDENTIFIER }

```

### 12.11.2.8 Issuer Alternative Name

This field is similar to the previous one for the issuer identity.

```
issuerAltName EXTENSION ::= {
  SYNTAX      GeneralNames
  IDENTIFIED BY id-ce-issuerAltName }
```

### 12.11.2.9 Subject Directory attributes

This field is used to carry optional directory attributes associated with the subject.

```
subjectDirectoryAttributes EXTENSION ::= {
  SYNTAX      AttributesSyntax
  IDENTIFIED BY id-ce-subjectDirectoryAttributes }
```

```
AttributesSyntax ::= SEQUENCE SIZE (1..MAX) OF Attribute
```

### 12.11.2.10 Basic Constraints

This field indicates if the subject of the certificate is a certification authority and how deep a certification path may exist through that path.

This extension shall appear in every CA Certificates. It will be used to prevent unauthorised entities to act as a CA.

```
basicConstraints EXTENSION ::= {
  SYNTAX      BasicConstraintsSyntax
  IDENTIFIED BY id-ce-basicConstraints }
```

```
BasicConstraintsSyntax ::= SEQUENCE {
  cA                      BOOLEAN DEFAULT FALSE,
  pathLenConstraint       INTEGER (0..MAX) OPTIONAL }
```

### 12.11.2.11 Name Constraints

This field indicates a namespace within which all subject names in subsequent certificates in a certification path shall be located.

```
nameConstraints EXTENSION ::= {
  SYNTAX      NameConstraintsSyntax
  IDENTIFIED BY id-ce-nameConstraints }
```

```
NameConstraintsSyntax ::= SEQUENCE {
  permittedSubtrees       [0] GeneralSubtrees OPTIONAL,
  excludedSubtrees        [1] GeneralSubtrees OPTIONAL }
```

```
GeneralSubtrees ::= SEQUENCE SIZE (1..MAX) OF GeneralSubtree
```

```
GeneralSubtree ::= SEQUENCE {
  base                    GeneralName,
  minimum                 [0] BaseDistance DEFAULT 0,
  maximum                 [1] BaseDistance OPTIONAL }
```

```
BaseDistance ::= INTEGER (0..MAX)
```

### 12.11.2.12 Policy Constraints

This field is only used in CA's certificate (i.e. not in end entity certificate). It is used to prohibit policy mapping or to require that each certificate in a path contain an acceptable policy identifier.

```
policyConstraints EXTENSION ::= {
  SYNTAX      PolicyConstraintsSyntax
  IDENTIFIED BY id-ce-policyConstraints }
```

```
PolicyConstraintsSyntax ::= SEQUENCE {
  requireExplicitPolicy   [0] SkipCerts OPTIONAL,
  inhibitPolicyMapping    [1] SkipCerts OPTIONAL }
```

```
SkipCerts ::= INTEGER (0..MAX)
```

### 12.11.2.13 Extended key usage field

This field is an extensions of the previous field `keyUsage`. It is used to define more purposes for which the certified public key may be used.

```
extKeyUsage EXTENSION ::= {
  SYNTAX      SEQUENCE SIZE (1..MAX) OF KeyPurposeId
  IDENTIFIED BY id-ce-extKeyUsage }
```

```
KeyPurposeId ::= OBJECT IDENTIFIER
```

### 12.11.2.14 CRL Distribution points

This field defines how CRL (Certificate revocation list) information can be obtained.

```
cRLDistributionPoints EXTENSION ::= {
  SYNTAX      CRLDistPointsSyntax
  IDENTIFIED BY id-ce-cRLDistributionPoints }
```

```
CRLDistPointsSyntax ::= SEQUENCE SIZE (1..MAX) OF DistributionPoint
```

```
DistributionPoint ::= SEQUENCE {
  distributionPoint [0] DistributionPointName OPTIONAL,
  reasons           [1] ReasonFlags OPTIONAL,
  cRLIssuer         [2] GeneralNames OPTIONAL }
```

```
DistributionPointName ::= CHOICE {
  fullName           [0] GeneralNames,
  nameRelativeToCRLIssuer [1] RelativeDistinguishedName }
```

```
ReasonFlags ::= BIT STRING {
  unused           (0),
  keyCompromise   (1),
  cACompromise    (2),
  affiliationChanged (3),
  superseded      (4),
  cessationOfOperation (5),
  certificateHold  (6) }
```

## 12.12 MHP certification procedures

This MHP certification procedures reference is a place holder for a document that will have to be written and agreed with the root CA that describe the administration of certificates.

As defined in section 12.5.6 on page 295, each leaf certificate shall contain the `organisationId` in the organisation name attribute. In case there are several root certificate entities, the certificate procedure shall ensure that the leaf certificate contain the `organisationId` defined in ETR 162 [10] which is owned by this organisation.

## 12.13 Plug-ins

The security aspects for the plug-in application are the same as for any other DVB-J application. The security aspects of the delegated application cannot exceed the permissions available to the plug-in.

So, the plug-in application will normally request the set of permissions required to support a delegated application. For a delegated application that follows the security model defined in this specification (e.g. a DVB-HTML application), the plug-in shall implement a security system to grant a sub-set of these permissions to the delegated application (see annex AF, "(normative): Plug-in APIs" on page 1328). For a delegated application whose security model does not follow this specification, the plug-in can optionally implement a private security system to manage a sub-set of these permissions to the delegated application.

## 12.14 Applications loaded from an interaction channel

Same as for applications delivered over the broadcast channel except where specified otherwise below.

### 12.14.1 Permission for application loading from the return channel

The following policy covers loading of applications over the interaction channel. It includes both initial loading to start and application and dynamic loading during the lifetime of an application. It covers both the code of an application and supporting data however loaded.

- The user enters a locator specifying the location of the AIT file into the navigator:

In this case the user implicitly gives permission for the connection to retrieve the AIT when they supply the locator.

Applications launched by this mechanism inherit limited rights to use the return channel. Where the protocol ID is "3" the application gets rights to access the files specified by the transport protocol descriptor.

Additional permissions can be granted by the normal permissions mechanism.

- A service has a broadcast AIT and the transport protocol for the application is type "3" so ALL files are provided by the interaction channel.

If the application launch is initiated by another application using the listing and launching API and it is necessary to open the interaction channel before the permissions of the launched application are evident then the permissions of the launching application are considered.

If the application launch is initiated from service selection (autostart application on a service) then the behaviour depends on policy in the MHP terminal.

NOTE: This behaviour may, for example, be controlled by a terminal user preference.

- A service has a broadcast AIT but a proportion of the files forming the application are provided via the interaction channel.

If the permission request file and the authentication infrastructure are available without opening the interaction channel then these can be used to determine if opening the interaction channel is suitably authorised.

- An application selects a service using a locator pointing to an AIT file delivered over the return channel to perform service selection

In this case the service selection succeeds only if the application has permission to do a service selection and the appropriate return channel access permission, as described in section 12.6.2.12.2, "Signed applications" on page 308.

If the permission request file is not available then, as above, the behaviour depends on policy in the MHP terminal.

## 12.15 Stored applications

Applications can be authenticated as describe in MHP 1.0.

Application authentication is modelled in two phases:

- a) At time of application storage all of the certificates authenticating the application must be valid (unexpired and unrevoked) for the application to be considered authenticated.
- b) At the time of application retrieval from storage if any certificate authenticating the application has been revoked then the application will be treated as failing authentication.

## **12.16 Inner applications and content embedded within other applications**

Inner applications shall inherit the set of permissions granted to the application in which they are embedded.

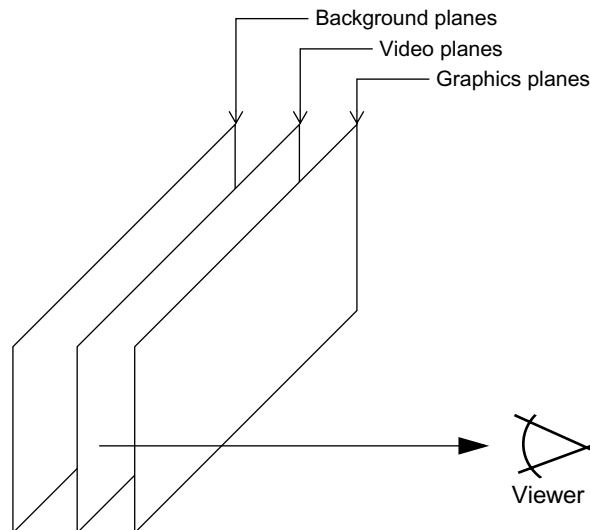


## 13 Graphics reference model

### 13.1 Introduction

The MHP provides tools to control the positioning of: video on an output device, interface components such as buttons and lists, as well as raw graphical primitives.

Each screen connected to an MHP has three planes which are, from back to front, a background plane, a video plane and a graphics plane.



**Figure 21 : Illustration of the different types of display planes**

The behaviour of the subtitle plane varies between implementations. API facilities are provided to allow applications to make the behaviour predictable. See 13.5, "Subtitles" on page 349.

An application is provided with a contiguous rectangular region of the graphics plane in which it can draw (see 13.3.3, "HAvi devices and AWT components" on page 341). An application can place video, interface elements and graphics inside its rectangle on the graphic plane.

An application can also control video outside of the AWT hierarchy on the video plane, and place still images or solid colour in the background plane.

The MHP specification enables terminals to support multiple applications at any one time, each of which can have a sub area of the screen to which it can draw. The specification enables the areas to overlap. However, the minimum required support for these features is profile specific. See annex G, "(normative): Minimum Platform Capabilities" on page 442.

#### 13.1.1 Interapplication interaction

If the presentations of different applications overlap, the areas obscured by other applications are clipped. Therefore where an application is translucent it will be blended with the video or background image behind it rather than being blended with another application.

## 13.2 General Issues

### 13.2.1 Coordinate Spaces

The MHP includes a number of coordinate systems for different purposes and includes the means to transform between these as needed:

- Input video space

This considers post upsampling MPEG pixels.

- Device space

Logical pixels in the various display devices. There may be different device spaces for the various device types (e. g. video and graphics).

- Normalised screen space

Normalised coordinates relative to the output (HScreen).

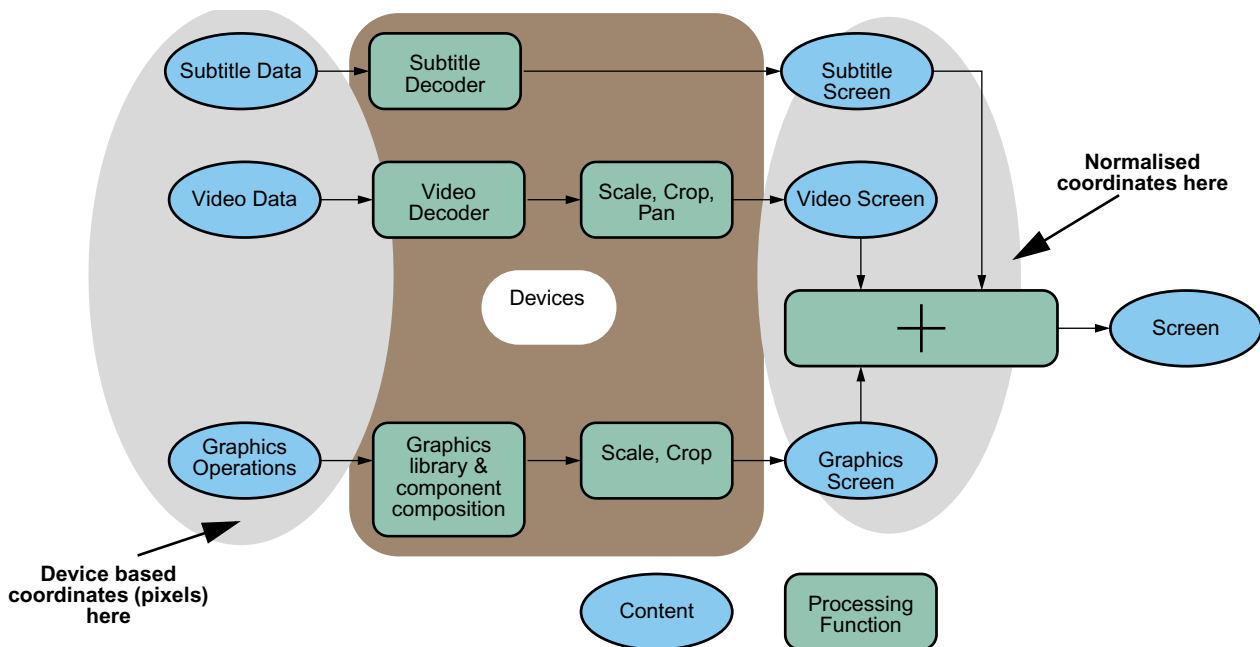


Figure 22 :

Various different interfaces provide access to different parts of the graphics and video systems using these coordinate spaces.

#### 13.2.1.1 Normalised screen space

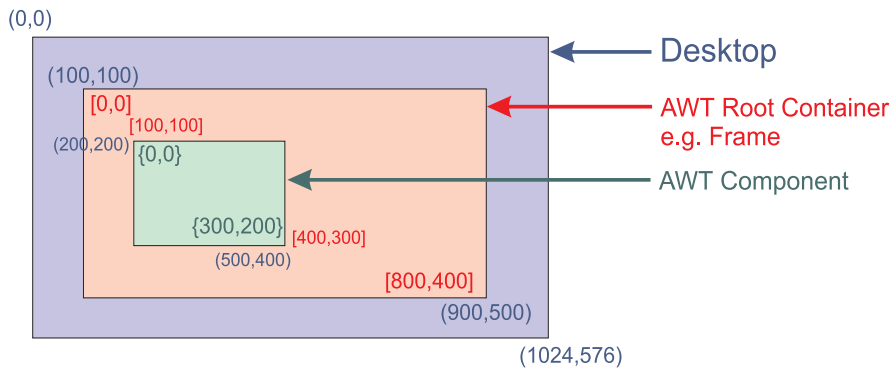
A normalised screen coordinate system supports references to positions and sizes in the video output from an MHP device without reference to any form of pixels.

This coordinate system describes the top left corner of the screen as  $\{0, 0\}$  and the bottom right corner of the screen as  $\{1, 1\}$ . This coordinate system is used in the positioning of graphics and video on the screen through a number of classes in the `org.havi.ui` package and the JMF control `org.dvb.media.VideoPresentationControl`.

The normalized coordinate system is given through the `HScreen`.

### 13.2.1.2 User space

The coordinate space for graphics used in java.awt is defined by applications through the creation of an `HGraphicsDevice`. The root container, an instance of the class `org.havi.ui.HScene` can be placed within the normalised coordinate system. The `HSceneTemplate` class allows applications to express requirements for their top level container. Instances of this class are used by `HSceneFactory` to return instances of an `HScene`.



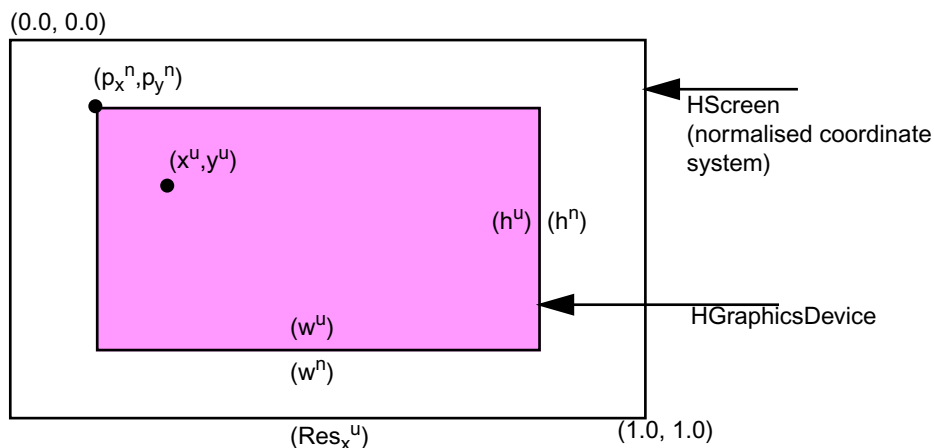
#### Coordinate Systems

- $(x,y)$  coordinate system of the Desktop
- $[x,y]$  coordinate system of the root container
- $\{x,y\}$  coordinate system of the component

**Figure 23 : AWT Coordinate system in a computer environment**

Figure 23 shows the AWT coordinate system in a normal computer environment. In an MHP device the `HGraphicsDevice` can be thought of being the desktop and the `HScene` as being the AWT RootContainer. Thus an `HScene` is placed within the coordinate system of the `HGraphicsDevice`. An `HScene` itself defines a new coordinate system which pixels are aligned to those of the `HGraphicsDevice` but the origin is translated (see figure 23).

The mapping between the user space and the normalised coordinate system is done given the size and location of the `HGraphicsDevice` in the normalised coordinate system and the resolution of the `HGraphicsDevice` in pixel.



**Figure 24 : Conversion between normalised and user coordinate systems**

$(x^n, y^n)$  = normalised coordinate system of `HScreen`

$(x^u, y^u)$  = coordinate system of the `HGraphicsDevice` in pixels

$x^{vu}$  = "virtual" user coordinate system in pixels relative to the `HScreen`

Assume the `HGraphicsDevice` has the origin of  $(p_x^n, p_y^n)$  with a dimension of  $(w^n, h^n)$  and has a pixel resolution of  $w^u \times h^u$  then the point  $(x^u, y^u)$  in normalised coordinates is given by

$$x^n = \frac{x^{vu}}{Res_x^{vu}} \quad y^n = \frac{y^{vu}}{Res_y^{vu}}$$

where

$$Res_x^{vu} = \frac{w^u}{w^n} \quad Res_y^{vu} = \frac{h^u}{h^n}$$

is the virtual resolution of the HScreen in (sub)pixel and

$$x^{vu} = p_x^{vu} + x^u \quad y^{vu} = p_y^{vu} + y^u$$

is the point (x,y) in the virtual coordinate space with

$$p_x^{vu} = p_x^n \cdot Res_x^{vu} \quad p_y^{vu} = p_y^n \cdot Res_y^{vu}$$

being the location of the HGraphicsDevice in the virtual coordinate system.

Given the Point ( $x^n, y^n$ ) the point ( $x^u, y^u$ ) is given by

$$x^u = \text{floor}[(x^n - p_x^n) \cdot Res_x^{vu} + 0.5]$$

and

$$y^u = \text{floor}[(y^n - p_y^n) \cdot Res_y^{vu} + 0.5]$$

Given a HGraphicsDevice which is full screen this simplifies to

$Res_x^{vu}=w^u$  and  $Res_y^{vu}=h^u$ ,  $x^{vu} = x^u$  and  $y^{vu}=y^u$  thus

$$x^n = \frac{x^u}{w^u} \quad y^n = \frac{y^u}{h^u}$$

and

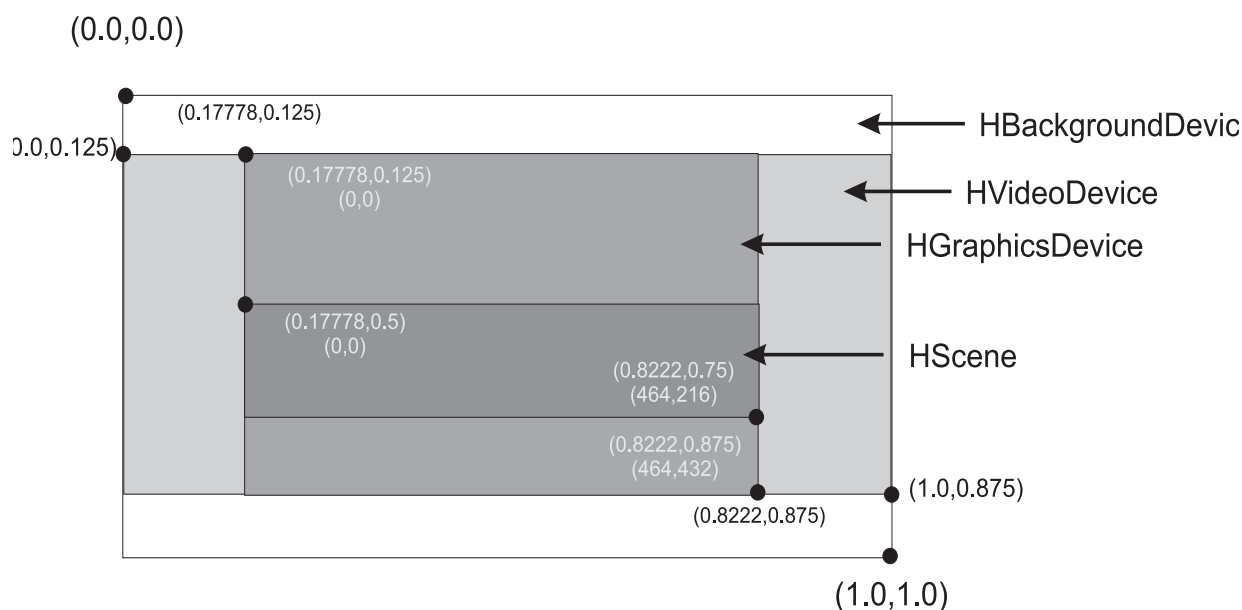
$$x^u = x^n \cdot w^u \quad y^u = y^n \cdot h^u$$

Within the above calculations precision equivalent to a float shall be used. Conversion to integer shall only be used when the result is a point in a pixel oriented co-ordinate space (e.g. user space).

The resolution of an HGraphicsDevice is specified using the constant HScreenConfigTemplate.PIXEL\_RESOLUTION with a Dimension on the setPreference(int, Object, int) method.

The constant HSceneTemplate.SCENE\_PIXEL\_RESOLUTION allows applications to define the dimension of an HScene in pixel of the HGraphicsDevice. The constant SCENE\_SCREEN\_RECTANGLE allows applications to define the position and size which the HScene should occupy on the screen in normalised coordinates.

The combination of these defines the transformation between graphics pixels and the video output from the MHP device concerned. Only a limited set of transformations are required to be supported. The `HGraphicsConfiguration` class contains methods to transform in both directions between graphics pixels and normalised coordinates.



**Figure 25 : Possible configuration of HAVi Devices**

Figure 25 shows a possible configuration of HAVi Devices. The `HBackgroundDevice` is configured to be full screen, the `HVideoDevice` to cover the area of (0.0, 0.125) to (1.0, 0.875).

When positioning video implementations may snap the video position to an adjacent line vertically (for example, to accommodate video and display field order) or an adjacent pixel horizontally (for example, to accommodate display chroma structure). The direction of "snapping" shall always be to minimise the error relative to the requested coordinate.

In figure 25 the `HGraphicsDevice` is configured to cover the area (0.17, 0.125) to (0.82, 0.875) with a pixel resolution of 464x432.

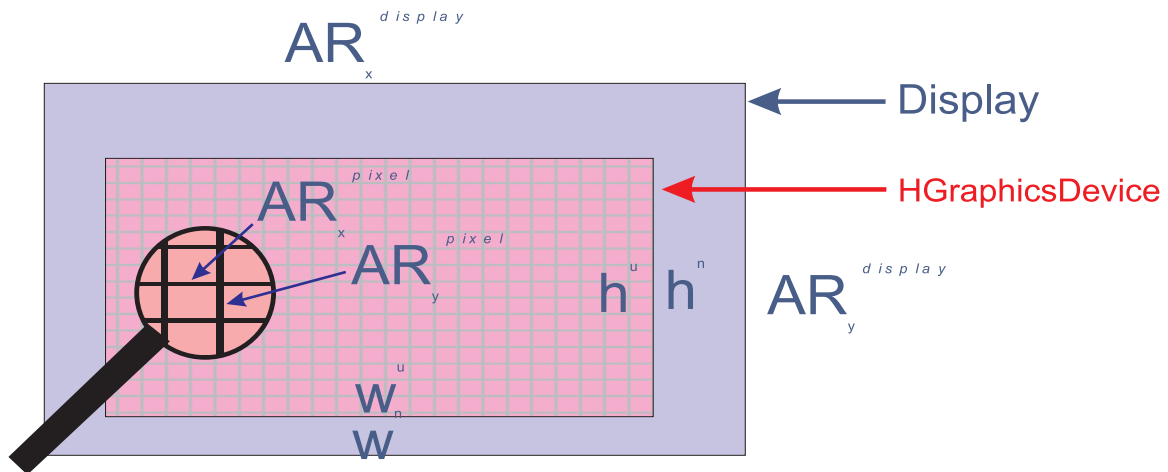
The `HScene` can be configured by setting the preference `HSceneTemplate.SCENE_PIXEL_RECTANGLE` with a `Rectangle` of (0, 216, 464, 261) or by setting the preference `HSceneTemplate.SCENE_SCREEN_RECTANGLE` with a `HScreenRectangle` of (0.17, 0.5, 0.64, 0.75).

In some implementations and/or configurations pixels in the `HGraphicsDevice` may not correspond to discrete physical pixels in the actual display device. For example, this may be the case when the `HGraphicsDevice` is emulated.

### 13.2.1.3 Pixel Aspect Ratio

A pixel orientated coordinate system does not say anything about the pixel aspect ratio. The pixel aspect ratio ( $AR_x^{pixel} / AR_y^{pixel}$ ) is defined by the aspect ratio of the display ( $AR_x^{display} / AR_y^{display}$ , 4:3 or 16:9), the area that is covered by the `HGraphicsDevice` ( $w^n, h^n$ ) and the pixel resolution of the `HGraphicsDevice` ( $w^u, h^u$ ). See figure 26

$$AR^{pixel} = \frac{AR_x^{pixel}}{AR_y^{pixel}} = \frac{AR_x^{display} \cdot w^n}{AR_y^{display} \cdot h^n} \cdot \frac{h^u}{w^u}$$



**Figure 26 : Calculating the Pixel Aspect Ratio**

Table 133 shows typical resolutions of a full screen HGraphicsDevice and the corresponding pixel aspect ratios for different display aspect ratios. The supported resolutions are defined in G, "(normative): Minimum Platform Capabilities" on page 442.

**Table 133 : Typical Resolutions and their pixel aspect ratio**

	4:3 Display	16:9 Display
Resolution for full screen HGraphicsDevice	Pixel Aspect Ratio	Pixel Aspect Ratio
720x576	16:15	64:45
768x576	1:1	48:36
1024x576	36:48	1:1

#### 13.2.1.4 Video space

The coordinate space for video is that defined by the input video signal after any scaling required by the platform (e.g. that required by ETR154). Scaling and clipping of video can be achieved using the JMF control `org.dvb.media.VideoPresentationControl`. The JMF control `VideoFormatControl` allows applications to query the various transformations being performed on video as part of its decoding and presentation.

## 13.3 Graphics

### 13.3.1 Modelling of the MHP display stack composition

The following sections describes the theoretical model of the MHP display stack. Unfortunately, certain real world constraints may apply see section 13.6, "Approximations" on page 350. The "Graphics, Video and Subtitle pipeline" is illustrated in figure 27.

Figure 27 shows the conceptual model of the "graphical content" pipelines from an applications point of view. There are four sources of graphical content: subtitles, background, video and application graphics, and these four sources are mapped in a controllable way onto single screen. The figure shows the conceptual processing functions, the content stages, and application control points inside the content pipelines.

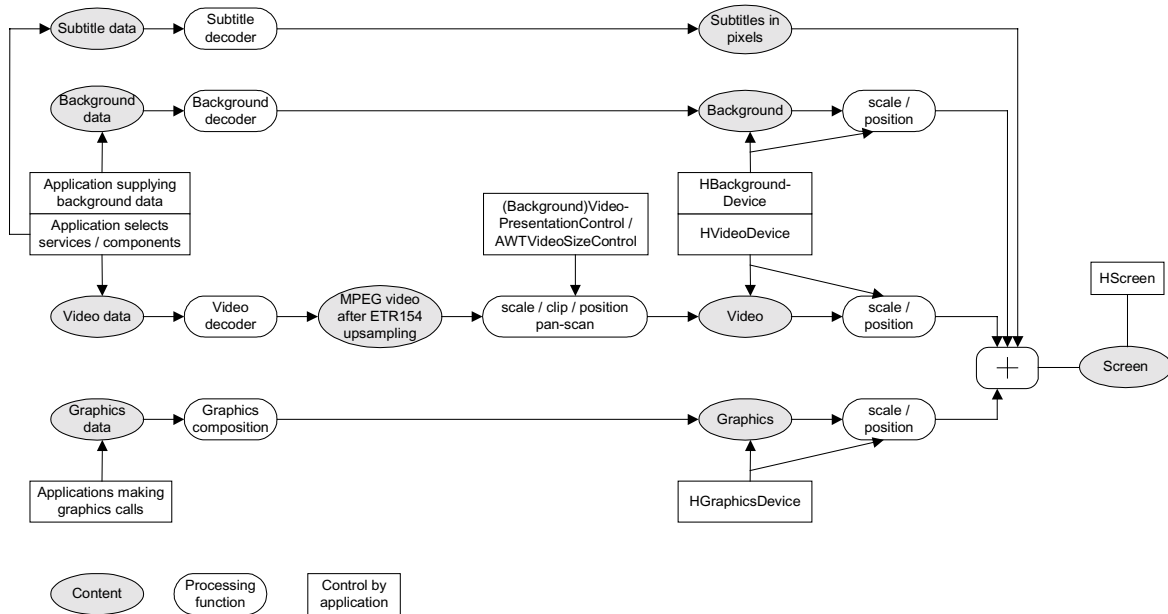


Figure 27 : Graphics, Video and Subtitle pipeline

The figure shows that applications have full control over the source of the content that enters the different pipelines. Furthermore the application may control clipping, scaling and positioning of the content at different stages in the different pipelines. Of course an application can only apply operations that are supported by the underlying system.

The clipping and positioning of subtitles follows the video. Behaviour of subtitles when video is scaled is platform dependent.

From the application author's point of view the behaviour of the graphics composition process has 3 elements:

- The AWT elements are composed following the traditional graphics model using Porter-Duff rules (see Porter-Duff [D]). The default rule used is the SRC rule.
- The background and video planes are composed using the Porter-Duff rule SRC\_OVER (Note that only alpha of 0 and 1 is used in the Video planes).
- The results are composed together using the SRC\_OVER rule (with the AWT results as the source and the HAVi results as the destination).

This is illustrated in figure 28.

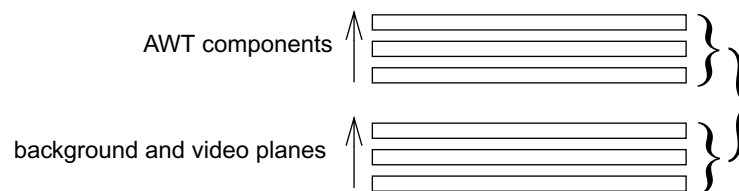


Figure 28 : Overview of AWT / HAVi plane composition

The HAVi stack has a single full screen `HBackgroundDevice` at the back. In front of this are an ordered set of zero or more `HVideoDevices`. Each of the video devices can occupy the full screen area or part of it. Any area that is not occupied behaves as transparent. The occupied areas (video pixels) are considered to be opaque and will be obscured by any video devices in front of them. Non active video devices are invisible.

Systems that support only a single video device that can display full screen (and possibly other partial screen video devices) should report the full screen capable device as the first (back most) of the video devices.

The composition rules in each group follow the traditional "painter's" algorithm i.e. composing the layers from back to front.

The Xlet AWT Root Component is transparent (as shown in figure 29) so by default the result of the HAVi video and background device composition is the background to any application graphics.

Video already running in the `HVideoDevice` will keep on playing when an application starts.

The stack is illustrated in figure 29.

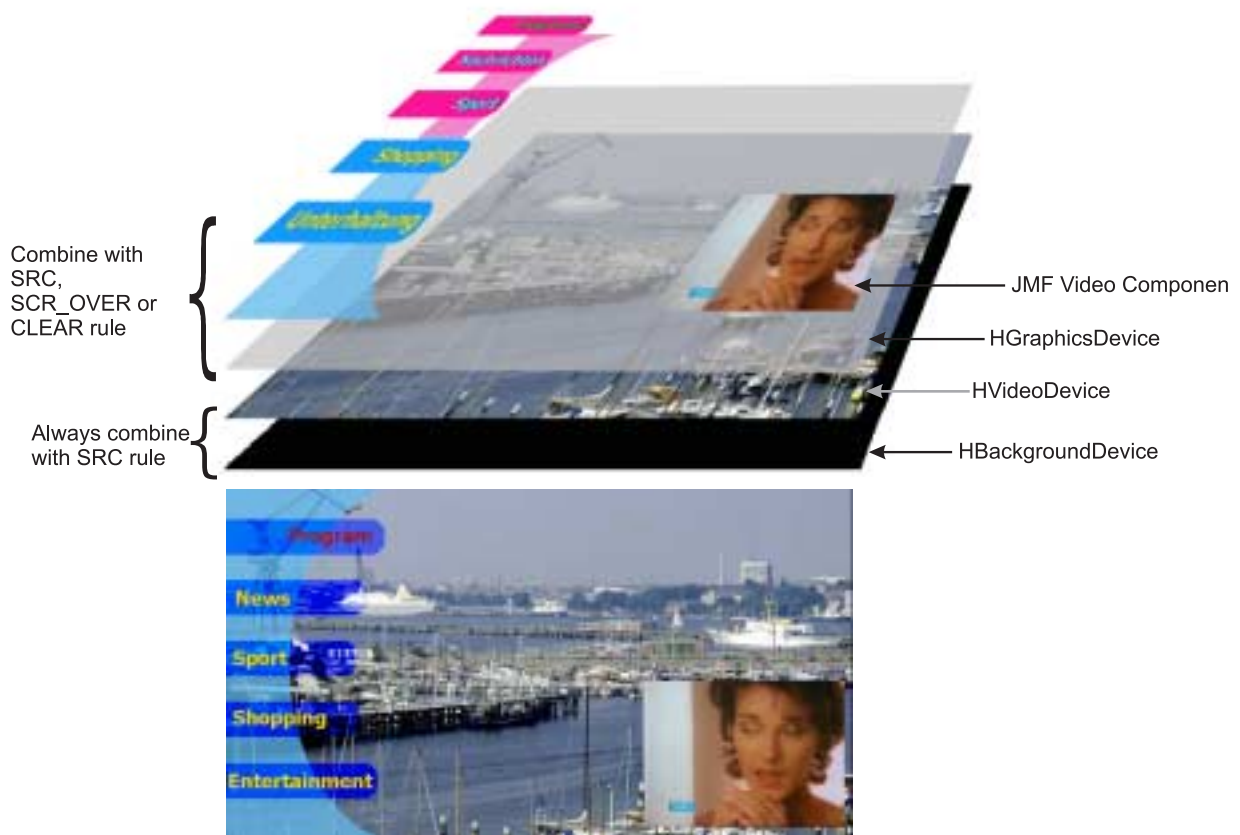


Figure 29 : The MHP display stack illustrated

### 13.3.2 AWT Reference Model in the MHP

In the AWT all graphics rendering of the lightweight components is done via the `java.awt.Graphics` class. When a component needs to be redrawn it issues a repaint command which gets passed from component to component from top to bottom until a heavyweight component is reached. Although there are no heavyweight components in the MHP the `HScene` can be thought of being similar to a heavyweight component. Thus in MHP devices the repaint command gets passed until it reaches the root container - the `HScene`. The repaint method of the `HScene` gets a `org.dvb.ui`.





MHP terminals are allowed to support only non overlapping HScenes. In implementations not supporting overlapping HScenes following behaviour shall be implemented:

- Overlapping at creation or size change through HSceneFactory:  
returns a null reference if "REQUIRED" is used. Returns best effort if "PREFERRED" is used.
- Overlapping later when sizes change by awt:  
Size does not change. Fails silently.

The MHP specification provides a general model for video output devices using the model found in the HAVi specification. A final output video signal is expressed through the `org.havi.ui.HScreen` class and is the result of adding a number of different video components.

- The output of one or more graphics decoders
- The output of one or more video decoders
- The output of any special decoders, e.g. a background

An abstraction of each of these decoders is represented by an instance of a class inheriting from `HScreenDevice` - `HGraphicsDevice`, `HVideoDevice` and `HBackgroundDevice` respectively. Each of these can potentially exist in a range of configurations which are exposed by instances of classes inheriting from `HScreenConfiguration` - `HGraphicsConfiguration`, `HVideoConfiguration` and `HBackgroundConfiguration` respectively. Where devices support multiple configurations, applications may construct and populate templates to define criteria to select between configurations. These templates are classes inheriting from `HScreenConfigTemplate`.

As well as the general features, some of these sub-classes provide support for specific features of the devices concerned. `HGraphicsConfiguration` supports loading of images and listing of fonts specific to that configuration. It also support conversion between various coordinate systems. `HVideoDevice` provides access to the source of the video currently being decoded (a `Locator`) and provides access to the decoder object for the video currently being decoded (a `javax.media.Player`). The `HBackgroundClass` provides a means to support MPEG I-frames through the `HStillImageBackgroundConfiguration` class.

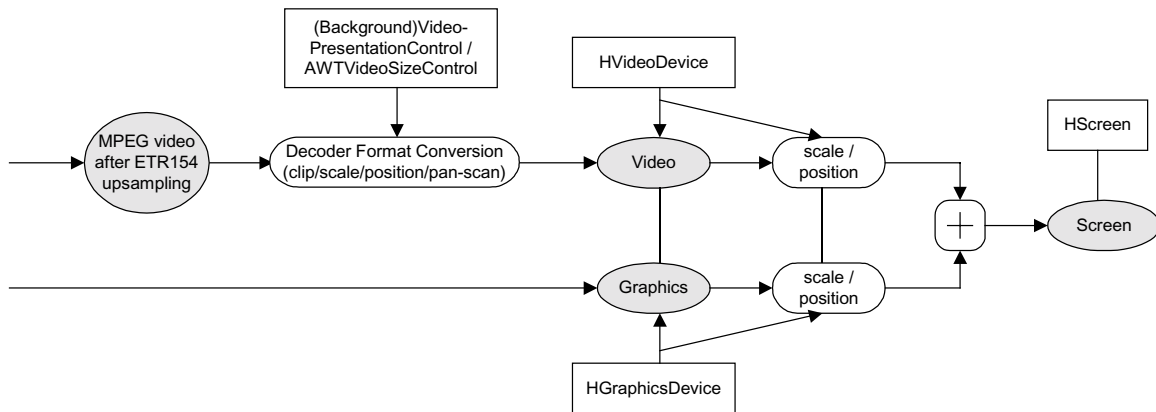
The method `HScreen.getCoherentScreenConfigurations ( HScreenConfigTemplate[] configs )` allows applications to express a common set of constraints for video, graphics and backgrounds and get back a coherent answer. In `HGraphicsConfigTemplate`, the constant `VIDEO_MIXING` allows applications to request configurations where graphics is super-imposed above video but without any requirement for pixels to be aligned. In `HScreenConfigTemplate`, there are constants to allow applications to ask for configurations as follows:-

- `VIDEO_GRAPHICS_PIXEL_ALIGNED` - video & graphics pixels are the same size and aligned
- `ZERO_VIDEO_IMPACT` - a graphics configuration must not change the existing video configuration
- `ZERO_GRAPHICS_IMPACT` - a video configuration must not change the existing graphics configuration

### 13.3.3.1 Video and graphics pixel aligned

The VIDEO\_GRAPHICS\_PIXEL\_ALIGNED relationship between the pixels in the HGraphicsDevice and the pixels in the HVideoDevice is shown in figure 31. Note that the relationship applies to the pixels in HVideoDevice after "Decoder Format Conversion" processing, and the pixels in HGraphicsDevice. As a result of this relationship the following constraint holds for the configurations of HGraphicsDevice and HVideoDevice that have aligned video and graphics pixels:

- the pixel aspect ratio of the pixels in both devices is equal.



**Figure 31 : Video and graphics pixel impact**

### 13.3.3.2 Zero graphics impact

Constants ZERO\_GRAPHICS\_IMPACT and ZERO\_VIDEO\_IMPACT can be used by applications to prevent changes to the HGraphicsDevice or the HVideoDevice respectively, in the case that changes are not intended. In general a change of the configuration of the HGraphicsDevice may lead to an automatic change of the configuration of, for example, the HVideoDevice (if the application is authorised to make this change), because restricted systems may not be able to deal with the two different configurations simultaneously. Therefore an application must specify ZERO\_GRAPHICS\_IMPACT or ZERO\_VIDEO\_IMPACT in the configuration template if it does not want to change the configuration of another device.

## 13.3.4 Composition

### 13.3.4.1 AWT paint rule

The normal AWT paint rules shall be followed. That is the root container (the HScene) is painted and then its components are painted recursively.

The observed *behaviour* shall be such that of each component drawing directly into the root component. Any use of temporary storage or double buffering shall not affect the final result of the AWT composition or its subsequent composition with the result of the video composition.

The process is shown in Figure 32. The numbers in the arrows indicate the chronological order of the painting (note the root container is painted first). If a container has multiple components they get painted in the order N, N-1, ..., 0 where N indicates the order components are added to the parent container. (See Documentation on `java.awt.Container` in JAE 1.1.8 API [31]).

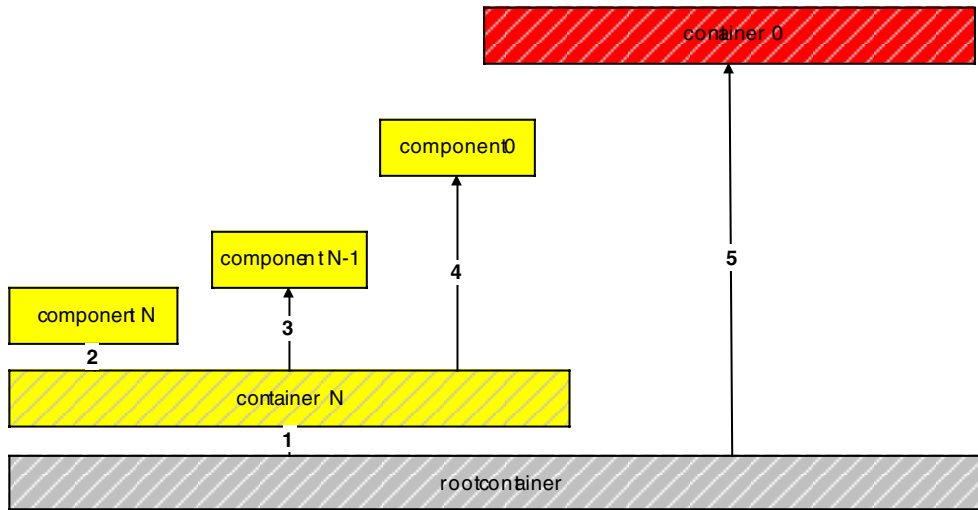


Figure 32 : Chronological order of painting

### 13.3.5 Composition Rules

#### 13.3.5.1 Components generally

By setting an appropriate `DVBAAlphaComposite` rule on a `DVBGraphics` and using translucent colours blending may be achieved. By repeated placement of components complex scenes can be described. In figure 33 the background picture (of the harbour) is representative of the video plane. The translucent grey rectangle is drawn into a transparent off-screen buffer using the SRC rule. The red circle is then drawn into this off-screen buffer using one of the 8 most common Porter-Duff operations. When the AWT rendering in the off-screen buffer is complete it is composited with the video using the SRC\_OVER rule.



Figure 33 : Summary of the Porter-Duff rules

### 13.3.6 Extensions to the AWT graphics capabilities

For graphics the MHP specification mainly uses AWT facilities defined in Java 1.1.x. However, certain extensions are made.

In order to allow compositing in the MHP the graphics capabilities of the JDK1.1 have been extended by providing the `org.dvb.ui` package.

The package consist of the classes `DVBGraphics`, `DVBAlphaComposite`, `DVBBufferedImage`, `DVBColor` and one Exception. See U, "(normative): Extended graphics APIs" on page 753.

#### 13.3.6.1 Graphics Objects in the MHP

The class `org.dvb.ui.DVBGraphics` extends the normal `java.awt.Graphics` class by adding support for alpha compositing.

In the MHP each platform-created graphics object shall be an instance of `org.dvb.ui.DVBGraphics`.

`DVBGraphics` and `DVBAlphaComposite` has a principle support of 8 different Porter-Duff compositing rules but not all rules have to be supported for all `DVBGraphics` Objects.

Different `DVBGraphics` Object could support different compositing rules. E.g. a `DVBGraphics` Object created using a `DVBBufferedImage` with an image type of `DVBBufferedImage.TYPE_ADVANCED` supports all 8 specified Porter-Duff rules, `TYPE_BASIC` supports only `SRC`, `CLEAR`, `SRC_OVER`. Application can query the available compositing rules using `DVBGraphics.getAvailableCompositingRules()`. When setting an unsupported compositing rule a `org.dvb.ui.UnsupportedDrawingOperationException` will be thrown.

The supported compositing rules are defined in G, "(normative): Minimum Platform Capabilities" on page 442.

The default compositing rule used by all graphics objects is `SRC`.

#### 13.3.6.2 Buffered Image

The class `DVBBufferedImage` in the package `org.dvb.ui` adds the support for an accessible, transparent Image which can be used e.g. for off screen buffers. Two different platform dependent sample models are supported by `DVBBufferedImage`. When doing compositing in the `TYPE_BASE` sample model approximations may be applied (using `SRC` instead of `SRC_OVER`, see figure 41 or by approximating the alpha, see G, "(normative): Minimum Platform Capabilities" on page 442) while in the `TYPE_ADVANCED` sample model the compositing rules set by the program will be used.

`TYPE_ADVANCED` is always a direct colour model while `TYPE_BASE` can be a CLUT based colour model.

#### 13.3.6.3 DVBColor

Note The general philosophy of this class has been to imitate the features of JDK 1.2 dealing with colour.

Unless explicitly specified otherwise, the internal implementation of the platform (e.g. `java.awt`) shall use the `org.dvb.ui.DVBColor` class instead of the `java.awt.Color` where technically possible (e.g. not the constructor for `java.awt.Color`). The class signatures shall not change. For example, where a method is specified to return `java.awt.Color` it shall return an instance of `org.dvb.ui.DVBColor`.

##### 13.3.6.3.1 Modified packed colour representation

The most significant byte of the integer representation of an RGB colour is defined to hold an alpha value as is illustrated in figure 34. This data type is used in various of the constructors and methods described in the API documentation. It is referred to in the API documentation as `TYPE_INT_ARGB`.

MSB				LSB			
31	24	23	16	15	8	7	0
Alpha		Red		Green		Blue	
7	0	7	0	7	0	7	0

Figure 34 : MHP colour format (`TYPE_INT_ARGB`)

## 13.4 Video

### 13.4.1 Component-based players and background players

Video is received by the MHP as an MPEG sequence of compressed frames, each of which contains a large number of picture elements (pels) arranged in rows and columns. The MHP decodes the MPEG stream to a presented stream which may be placed either in the video plane, or in a component in the graphics plane.

These two different ways of presenting video result in having two different kinds of JMF players, a background JMF player and a component-based JMF player.

A component-based JMF player plays video inside an AWT component, and the video inside that component is positioned and scaled by positioning and resizing the component. The video is always scaled to the full size of the component. Support for component-based players is not mandatory in all profiles.

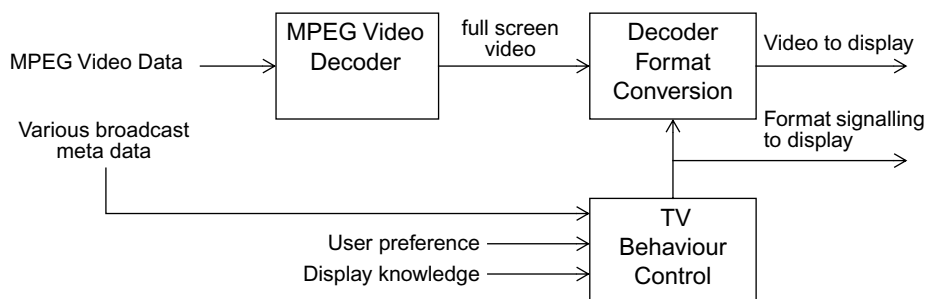
Support for background players is mandatory for broadcast streaming formats (see 7.2, "Broadcast streaming formats" on page 66).

Background JMF players play video in the video plane, outside and independent of any AWT component hierarchy.

### 13.4.2 Modelling MPEG decoding and presentation pipeline

Figure 35 illustrates the underlying format conversion control process in a TR 101 154 [9] compliant SD digital receiver. In this model the video decoder produces "full-screen" video from the MPEG data (i.e. the decoder resolves any sub-sampling in the MPEG broadcast). Subsequent "Decoder Format Conversion" adapts this for the display device taking account of:

- broadcast meta data (aspect ratio, pan/scan and active format description)
- display knowledge (4:3 or 16:9, and resolution)
- user preferences (e.g. display wide screen in a letterbox)



**Figure 35 : Format control in TV mode**

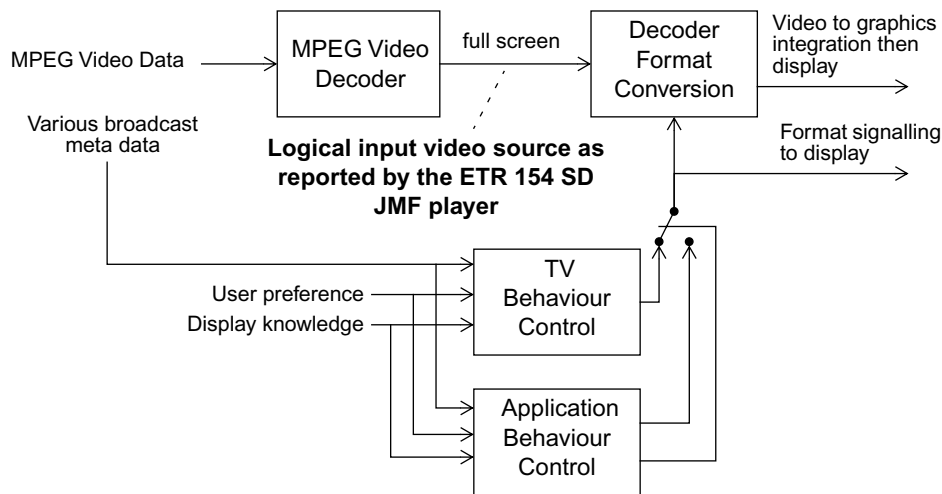
If appropriate, the video that is sent to the display is accompanied by the proper WSS or SCART signalling to indicate the format of the video that is being sent.

Note that the display device may do its own Decoder Format Conversion on top of the Decoder Format Conversion that the MHP device does. This is beyond the control of the MHP device.

The JMF players in this version of the MHP specification are "DVB ETR 154 Standard Definition" players and so act as if they are taking the full screen output of the MPEG Video Decoder as their logical input video source, as is illustrated in figure 36. In addition, there are two alternative sources of control for the "Decoder Format Conversion" process:

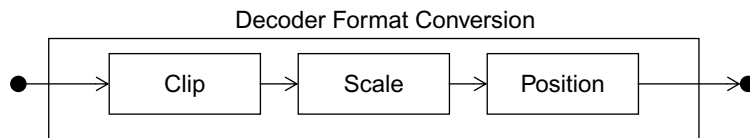
- Conventional TV format control behaviour (as in figure 35)
- Application format control behaviour

The selection between these behaviours is under the control of the application. Before and after the existence of an application the behaviour is that for a conventional TV. During application execution the default behaviour of the JMF player's decoder format conversion shall be the conventional TV behaviour.



**Figure 36 : Format control in the presence of a JMF player**

The Decoder Format Conversion consists of three steps that are performed on the full screen input video, which may have been up-sampled by the MPEG video decoder to become full screen. As illustrated in Figure 37, these steps are clipping, scaling and positioning.



**Figure 37 : Reference model for Decoder Format Conversion**

An application can query the implemented capabilities of each step of the Decoder Format Conversion using the `org.dvb.media.VideoPresentationControl` ("VideoPresentationControl" on page 609). For instance, it can query the supported scaling factors.

An application can also set up the Decoder Format Conversion steps atomically by using an `org.dvb.media.VideoTransformation` object ("VideoTransformation" on page 614) that encapsulates the clipping, scaling, and positioning parameters. An application can get a number of pre-defined video transformations that correspond with standard Decoder Format Conversions like 16:9 letterboxing in a 4:3 display. It can either use these video transformations directly to set the Decoder Format Conversion, or it can change one or more parameters of the transformation before setting it. The API also offers support for querying the current video transformation.

### 13.4.3 Coordinate Spaces

The input to the Decoder Format Conversion block is always full screen video. If necessary, the MPEG video decoder block performs up-sampling as required by ETR154 to get full screen video.

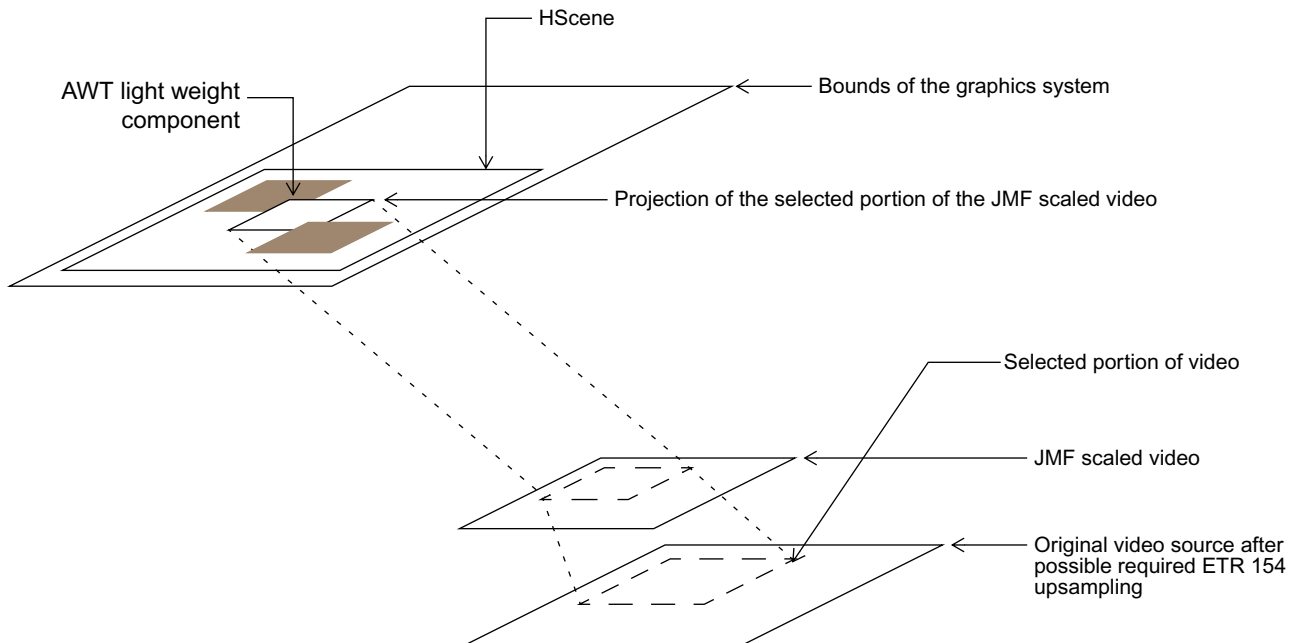
An application expresses the video clipping in terms of the pixel-based coordinate space of the full screen video. For 50Hz SD video this is always a 720x576 raster.

Positioning of the video is expressed in the normalised coordinate space for background JMF players. For component-based JMF players, the position of the video component is expressed in the pixel-based device coordinate space (i.e., a video component is positioned like any other AWT component).

### 13.4.4 Video components

A scaled portion of a video can be presented as a component within the AWT hierarchy. The controls that influence this presentation are parts of AWT and JMF.

Video components are treated just like any other component. However, it shall be noted that pixels within the video have opaque colours. The mapping to the source video is provided by the video presentation control (see N, "(normative): Streamed Media API Extensions" on page 574) which allows an arbitrary portion of the video to be placed within the AWT hierarchy.



**Figure 38 : Introducing video into the AWT component stack**

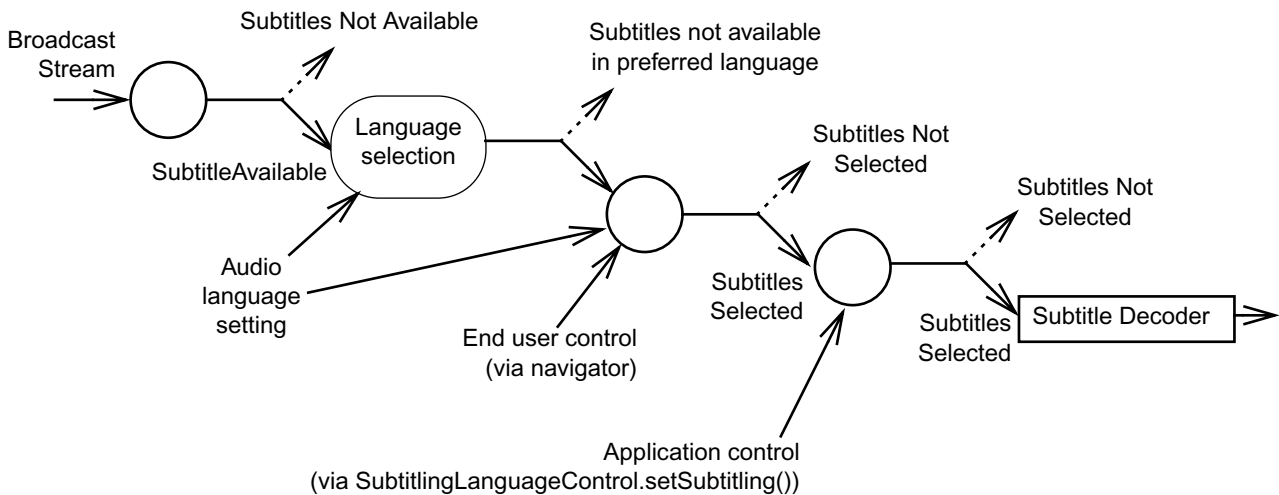


## 13.5 Subtitles

### 13.5.1 Language and presentation setting

The following reference model shows how the presentation of subtitles is controlled. The selection of the subtitles language depends by default on the user's preferences, the audio language and can be overridden by the application. The end user can set the subtitles on or off where the default depends on the preferences. The application can override all this and switch the subtitles off even if the user has set them on.

The figure below illustrates the decision procedure:



**Figure 39 : Determining subtitling language and presentation setting**

The DVB-J API includes a control (see "[SubtitlingEventControl](#)" on page 601) that the application can use to get notified of the state changes in the availability and presentation status of the subtitles. This corresponds to the status of the left hand side input and the right hand side output in the diagram.

The `org.davic.media.SubtitlingLanguageControl` allows the application to query the currently set subtitling language, override the default language setting and switch the subtitles off or let them be end user controlled. This corresponds to the language selection phase and the application controlled switch in the diagram

### 13.5.2 Relation to graphics

Ideally subtitles should be presented on top of the video plane but below the graphics plane(s). However, MHP terminals conforming to this specification are only required to support subtitles in areas where they are not overlapped by application graphics (i.e. by an `HScene` for a DVB-J application). The behaviour if the subtitles overlap with application graphics is non deterministic. Therefore, when presenting application graphics on screen, the application should either turn subtitles off or the broadcaster shall coordinate the use of screen area between subtitling and application graphics so that they will not overlap.

The subtitling plane is full screen and allows the subtitles to be positioned anywhere on screen. The positioning of the subtitling texts is part of the subtitling stream content and is fully controlled by the broadcaster.

### 13.5.3 Coordinate Spaces

Subtitles are decoded into a plane with the same coordinate system and position as the `HVideoDevice`.

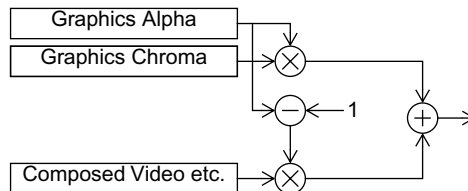
## 13.6 Approximations

### 13.6.1 Approximations in composition

The MHP specification references the Porter-Duff rules for composition (see [Porter-Duff \[D\]](#)). The set of operations required is profile specific in addition the implementation of the compositions may be approximated in some profiles.

#### 13.6.1.1 Implementation of modes

Typical implementations have a hardware blending process that blends the graphics plane over the video using SRC\_OVER (see [figure 40](#)). This places certain practical limitations on the purity of the display model.



**Figure 40 : Typical current technology implementation**

The SRC rule simply requires that the colour and alpha characteristics of the new pixels replace those previously present.

HVideoComponents are treated as having an alpha of 1 so whether SRC or SRC\_OVER is used when placing the video component the effect is that the video completely replaces anything previously drawn.

##### 13.6.1.1.1 Graphics directly over video

When drawing graphics directly over a video component:

- The effect of SRC\_OVER mode is as expected as the alpha value of the drawn graphics is used to control blending of the graphics with the video.

##### 13.6.1.1.2 Graphics over other graphics

###### 13.6.1.1.2.1 SRC

If (non-MPEG) graphics are painted over other (non-MPEG) graphics using SRC mode the result is as expected.

###### 13.6.1.1.2.2 SRC\_OVER

If (non-MPEG) graphics are painted over other (non-MPEG) graphics using SRC\_OVER mode the result will be implementation dependant. [Figure 41 on page 351](#) illustrates a variety of implementations of SRC\_OVER graphics to graphics blending.

###### 13.6.1.1.2.3 CLEAR

If (non-MPEG) graphics are painted over other (non-MPEG) graphics using the CLEAR mode the result is as expected. This operation is the same as using a source with alpha=0 and the SRC rule.

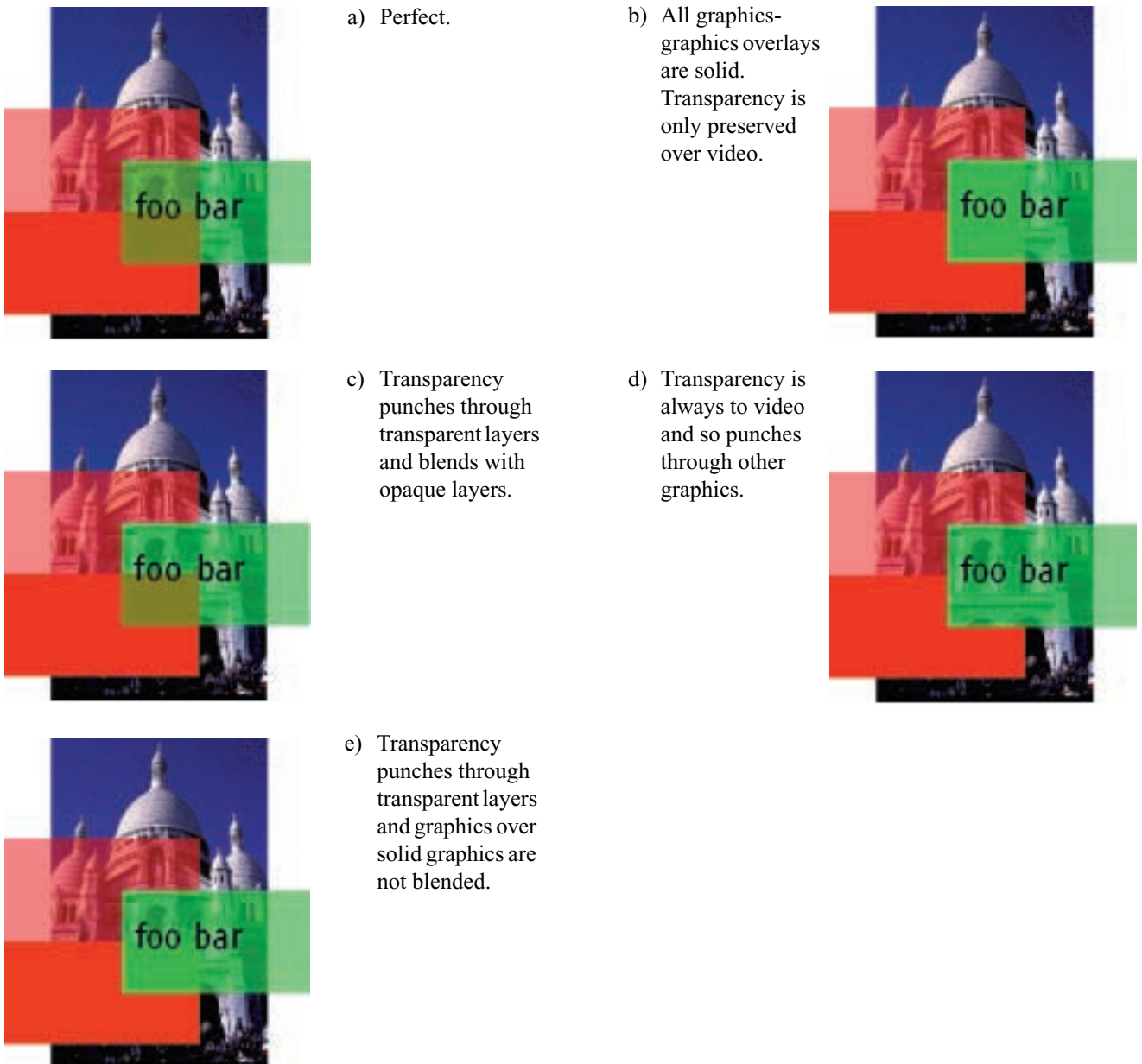
- [a] Shows the logically correct result where the green and red areas mix to produce intermediate colours.

This implies a graphics to graphics blending process, it also implies a large gamut in both the chroma and alpha channels. This may not be practical in many early implementations.

The following cases illustrate simplifications of the blending scheme. These should not be considered equally good approximations:

- [b] Here the graphics alpha is preserved only when it is drawn directly over a video component. When drawn over another (non-MPEG) graphic the alpha facets of the colours are considered to be 1.

- [c] Here the source graphics alpha is preserved when it is drawn over a video component even if there is an intermediate semi-transparent graphic. Where the source is over opaque graphics then a graphic to graphic blend is implemented.
- [d] Here the source graphics alpha becomes the hardware mixing alpha regardless of what has been drawn previously over the MPEG image.
- [e] Here the source graphics alpha becomes the hardware mixing alpha in areas where the alpha is already less than 1. Where the alpha is currently 1 (i.e. over opaque graphics) the alpha remains 1.



**Figure 41 : Implementations of blending**

### 13.6.1.2 Approximation of alpha

The precision of implementation of alpha depends on the `DVBGraphics` object concerned. The minimum requirements are specified in G, "(normative): Minimum Platform Capabilities" on page 442. The actual colour used for a given colour can be queried using `org.dvb.ui.DVBGraphics.getBestColorMatch()`.

### 13.6.1.3 Approximation of colour

Logically the colour model is a "true colour" one. However, colour approximation is allowed as described in G.1.5, "Colour capabilities" on page 443.

## 14 System integration aspects

### 14.1 Namespace mapping (DVB Locator)

An extended format of the DAVIC DVB URL ([DAVIC 1.4.1p9 \[3\]](#)) shall be used for addressing DVB-SI entities as well as files within object carousels. This extension of the DAVIC locator is backwards compatible with both the original DAVIC locator as well as the UK DTG extension ([UK MHEG Profile \[B\]](#)). The main extensions are support for multiple component tags for specifying a subset of the components of a service, and a specified way of referencing files in an object carousel within a service.

Using the same informal notation as used above, the following locator formats shall be used:

```
dvb://<original_network_id>[.<transport_stream_id>][.<service_id>[.<component_tag>{&<component_tag>}][;.<event_id>]][/<path_segments>}
```

or

```
dvb://'<textual_service_identifier>' [.<component_tag>{&<component_tag>}][;.<event_id>]][/<path_segments>}
```

A more formal specification of the DVB URL expressed in BNF (as used in [RFC 2396 \[41\]](#)) is presented below:

**Table 134 : DVB URL syntax**

dvb_url	= dvb_scheme ":" dvb_hier_part
dvb_scheme	= "dvb"
dvb_hier_part	= dvb_net_path   dvb_abs_path
dvb_net_path	= "/" dvb_entity [ dvb_abs_path ]
dvb_entity	= dvb_transport_stream   dvb_service   dvb_service_component
dvb_transport_stream	= original_network_id "." transport_stream_id
dvb_service	= dvb_service_without_event [ dvb_event_constraint ]
dvb_service_component	= dvb_service_without_event "." component_tag_set [ dvb_event_constraint ]
dvb_service_without_event	= original_network_id "." [ transport_stream_id ] "." service_id   "'" textual_service_identifier "'"
component_tag_set	= component_tag *( "&" component_tag )
dvb_event_constraint	= ";" event_id
original_network_id	= hex_string
transport_stream_id	= hex_string
service_id	= hex_string
component_tag	= hex_string
event_id	= hex_string
hex_string	= 1*hex
hex	= digit   "A"   "B"   "C"   "D"   "E"   "F"   "a"   "b"   "c"   "d"   "e"   "f"
digit	= "0"   "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"
dvb_abs_path	= "/" path_segments
(path_segments as defined in <a href="#">RFC 2396 [41]</a> )	

It should be noted that this syntax is fully compliant with the generic syntax of URIs as specified in [RFC 2396 \[41\]](#) and uses the registry-based naming authority version of that. Furthermore, all generic definitions specified in [RFC 2396 \[41\]](#) shall be valid for the DVB URL as well (e.g. escaping of special characters within file names, etc.).

[RFC 2396 \[41\]](#) defines methods for path segments to include parameters (introduce with a semicolon character ";"). This specification currently makes no use of such parameters. Implementations conforming to this specification shall ignore any such parameters to ensure compatibility with future specifications.

#### 14.1.1 dvb\_entity = dvb\_service

When a path is present in a URL where the dvb\_entity part identifies a DVB service, the path references an object in an object carousel within the service. If the dvb\_service\_component element is not present there shall only be one Object Carousel in the DVB service.

### 14.1.2 dvb\_entity = dvb\_service\_component

When a path is present in a URL where the `dvb_entity` part identifies one component of a DVB service and that component carries an object carousel stream, the path references an object in an object carousel whose "root" (i.e. DSI message) is sent within that component. In this case the component tag set shall only contain one element.

The semantics when the path is present in URL where the `dvb_entity` part identifies something else than the two cases described above are not specified in this specification.

### 14.1.3 dvb\_hier\_part = dvb\_abs\_path

When the `dvb_net_path` part is missing and only the `dvb_abs_path` is present, the URL refers to a file in a default object carousel within the current service. The "current" service is dependent on the usage context.

### 14.1.4 dvb\_abs\_path

The following restrictions apply to the `dvb_abs_path` part of a name:

- The total length of pathnames, separators and filename shall be less than or equal to 254 bytes long.
- The following characters are not allowed in filenames and pathnames: character null (0xC080), byte zero.
- The encoding of the filename is in UTF-8 (see 7.1.5 on page 66).
- The directory separator character (i.e. Java's `path.separator` property) shall be a slash character (0x2F).
- An absolute filename starts with a slash character (as indicated in the BNF above).

## 14.2 Reserved names

File names starting with the characters "dvb." are reserved for use as "well known" files defined in this or future specifications.

Authors shall not use file names with this form to avoid possible collision with standards defined files.

## 14.3 XML notation

These rules shall apply to the processing and encoding of all the files where XML is used as an encoding format in the MHP.

Rules for encoding of the XML formatted files:

- the file shall be a well-formed XML document (but not necessarily valid against the DTD specified in this version of this specification). Here 'well-formed' and 'valid' are used as defined in the XML 1.0 (see XML 1.0 [65]) specification.

NOTE: The remark on validity is included, because it is possible to be valid only relative to one DTD. Valid documents relative to a DTD specified in a later version of this specification would not be valid relative to the DTD specified in this specification - however, the rules defined here intend to provide this future-proofness and allow terminal implementations compliant with this specification to be able to process files that may be encoded according to a later version of this specification.

- the XML files may contain the XMLDecl item ("<?xml ... ?>" tag) in the prologue in the beginning of the file
- all the XML files shall be formatted using the UTF-8 character encoding which is the default used in XML
- the possible XMLDecl item in the beginning of the file shall not contain an 'encoding' attribute specifying another encoding than UTF-8
- if the XMLDecl item is included in files conforming to this specification, it shall indicate XML version 1.0
- the XML file shall contain a document type declaration ("<!DOCTYPE ...>" tag) where the Name is the same as the name of the root element
- the document type declaration shall contain an ExternalID item with the "PUBLIC" identifier and both a PublicLiteral and a SystemLiteral. This specification specifies the PublicLiteral that shall be used to identify the document types defined in this specification. This specification specifies a SystemLiteral that can be used for identifying a location where the DTD can be retrieved. The SystemLiteral included in the document type declaration shall point to a location where the DTD can be obtained via the Internet using the HTTP protocol as specified in this specification.
- the PublicLiteral is used for identifying the type of the file. For document types specified in this specification, the PublicLiteral shall have the following syntax:

```
"-//DVB//DTD " <document type> " " <version_number> "-//EN"
```

where:

<document type> has the following syntax:

```
<document_type> = letter letters
letters = " " | letter letters
letter = uppercase_letter | lowercase_letter | space
uppercase_letter = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H"
                  | "I" | "J" | "K" | "L" | "M" | "N" | "O"
                  | "P" | "Q" | "R" | "S" | "T" | "U" | "V"
                  | "W" | "X" | "Y" | "Z"
lowercase_letter = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h"
                  | "i" | "j" | "k" | "l" | "m" | "n" | "o"
                  | "p" | "q" | "r" | "s" | "t" | "u" | "v"
                  | "w" | "x" | "y" | "z"
space = " "
```

<version\_number> has the following syntax:

```
<version_number> = major_version "." minor_version
major_version = digit digits
minor_version = digit digits
digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
digits = " " | digit digits
```

The `<document_type>` part of the `PublicLiteral` is used as an identifier of the document type. When future versions of this specification specify newer, backwards compatible versions of the document type, the `<document_type>` part shall not be changed and the `<version_number>` part shall be changed to a new version number unused in a previous version of this specification for that document type.

- the XML file shall be valid relative to the DTD identified in the `ExternalId` document type declaration. Here 'valid' is used as defined in the XML 1.0 specification.
- the document type declaration shall not contain a declaration part in square brackets ('[' and ']')
- where the XML type PCDATA is used in XML elements and it is specified that this contains a string (for example a file name), these strings shall not contain '<', '&' or '>' characters that might be mistaken as XML tags or references of the markup.

NOTE: Strings where these characters must be allowed should be specified to be encoded as CDATA, leading to a more complex notation but allowing those symbols to be used.

- the file shall include only tags and attributes that are defined in this specification or a later version of this specification
- the file shall not include XML entity declarations ("`<!ENTITY ...>`" tags)
- the file shall not include XML character or entity references (references starting with '&' character)
- the file shall not include XML processing instructions, except optionally the "`<?xml ...?>`" XMLDecl item
- the file may include XML comments ("`<!-- ... -->`" strings), but not within elements that are specified as PCDATA containing strings to be encoded as defined in this specification

Rules for processing of the XML formatted files in the MHP terminal:

- the parser shall use the `PublicLiteral` in the document type declaration in the XML file ("`<!DOCTYPE ...>`" tag) for identifying the type of the file.
- The `PublicLiteral` in the document type declaration identifies the version of the DTD that is used for this file. There is no requirement for the parser to try to fetch that DTD file using the URL defined by the `SystemLiteral`. It is an implementation option for the parser to retrieve the DTD from that URL. If the DTD is unavailable from that URL, then the behaviour shall be platform dependent.
- The parser shall accept files that have a different version number in the `PublicLiteral` than the one specified in this specification for the given file type. These are probably files encoded according to a later version of this specification. From those files, the parser shall parse, recognize and handle all those elements and attributes that are part of the DTD included in this specification.
- the parser shall ignore all XML elements (start tag, end tag, and possible string between them) that are not specified in the DTD included in this specification.



NOTE: This allows extending the DTDs in the future in a future proof manner where existing terminals ignore all the elements introduced in later versions of this specification

- the parser shall ignore such attributes of XML tags that are not specified in the DTD included in this specification
- the parser shall ignore XML comments encoded as defined in the XML 1.0 specification
- the parser must accept empty XML elements specified in this specification both in their start-tag and end-tag form as well as in the empty element tag form (e.g. '<tuning value="true"></tuning>' may be used as well as '<tuning value="true"/>')
- rules for evolving the specification must ensure that the encoding of the file will always be maintained backwards compatible when these rules are followed (i.e. later versions of this specification may add new XML tags and new attributes to existing XML tags, but may not change the semantics of the existing elements)
- if the encoding of the file violates the rules defined for the encoding above, the behaviour of the parser can be platform dependent, including the possibility that the parser may completely discard such files and the system may behave as if the file is not present at all.

## 14.4 Network signalling

The behaviour of MHP terminals when receiving incorrect transmissions is not specified. MHP terminals may implement whatever strategy they like for this situation. It is an allowed implementation choice to pass values from the network straight through to applications without checking them for correctness. Hence API calls which are specified as returning a specific piece of information may not return a valid piece of information if the original information in the network is wrong.

## 14.5 Text encoding of application identifiers

Where an [organisation\\_id](#) or [application\\_id](#) is encoded in textual form it shall be encoded as follows:

- a hexadecimal representation of the value
- lower case letters
- no extra leading zeros (as would be produced by Integer.toHexString)

Where both an [organisation\\_id](#) and [application\\_id](#) are combined into an application identifier, they will be represented as a single hexadecimal number using the previously described encoding with the [organisation\\_id](#) as the most significant bits and the [application\\_id](#) as the least significant bits.

## 14.6 Reserved names for persistent storage

File names are restricted to `persistentstoragename` as specified in the following BNF:

```

lowalpha = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" |
           "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" |
           "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"

upalpha  = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" |
           "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" |
           "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"

alpha    = lowalpha | upalpha

digit    = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
           "8" | "9"

punct    = "_"

persistentstoragename = alpha | digit | punct

```

## 14.7 Files and file names

The MHP specification defines how applications can use the names of files in order to access content held in files. It is intentionally silent about the file systems and file system namespaces of MHP terminals except as defined below.

- When an MHP application starts, the filesystem where that application is carried will be mounted into the file system namespace of the MHP terminal concerned. For a DVB-J application 11.5.1, "Broadcast Transport Protocol Access API" on page 255 defines that creating a new instance of `java.io.File(". ")` will result in a reference to the base directory of the application. This base directory may be a sub-directory within this filesystem.
- MHP applications which have requested the right to access persistent storage, and had this right granted, are allowed to access the persistent file namespace. For DVB-J applications, the top level directory of this namespace is obtainable from the system property, `"dvb.persistent.root"`.
- MHP applications may have the ability to mount additional filesystems into the file system namespace of the MHP terminal concerned. DVB-J applications are allowed to use the `attach()` method on the `org.dvb.dsmcc.ServiceDomain` class in order to attach an object carousel as an additional file system. In all other methods, using a DVB locator including the `dvb_abs_path` part of the name part of the syntax shall not mount the specified object carousel file system.
- Conformant MHP applications shall not attempt to access files or file systems outside what is allowed by this specification. The consequences should they attempt to do this are undefined and implementation dependent. Platforms are allowed to choose to limit the access rights of DVB-J applications through use of platform security mechanisms, e.g. `java.io.FilePermission`.
- References to content carried in files shall either be done using names of files encoded in text or using "file:" URLs as defined in RFC 1738 [67]. File names encoded in "dvb:" URLs shall be transformed to "file:" URLs before use. For DVB-J applications, file names shall be encoded in Java String objects and "file:" URLs shall be encoded in instances of `java.net.URL`.
- Stand-alone interaction channel applications shall see a file system where the current directory is derived from the contents of the "AIT file" (see 10.4.9 on page 213). Unless the MHP terminal is certain of the complete contents of a directory in this file system, all attempts to list such contents shall fail as if an I/O error had occurred in the underlying system.

## 14.8 Locators and content referencing

The table below lists the types of entity which may be addressed by locators in this specification and defines the text representation for each entity. This specification does not require support for addressing any other type of entity in an MHP system by locator or URL.

**Table 135 : Addressable entities, locators and their text representation (Sheet 1 of 2)**

Entity	Text Representation
Transport stream	DVB locator including "dvb_transport_stream" element
Network	No standardised text representation
Bouquet	No standardised text representation
DVB Service	DVB locator including "dvb_service" element
Generic Service	No standardised text representation unless also a DVB service
DVB Event	DVB locator including "dvb_service" element and "dvb_event_constraint" element
MPEG Elementary Stream	DVB locator including "dvb_service_component" element
File	"file:" URL as defined in RFC 1738 [67] (note 1) DVB locator including "dvb_abs_path" element (note 2) "http:" URL as defined in RFC 2616 [40]. "https:" URL as defined in RFC 2818 [98].

**Table 135 : Addressable entities, locators and their text representation (Sheet 2 of 2)**

Entity	Text Representation
Directory	"file:" URL as defined in RFC 1738 [67] (note 1) DVB locator including "dwb_abs_path" element (note 2) "http:" URL as defined in RFC 2616 [40]. "https:" URL as defined in RFC 2818 [98].
Drip feed decoder	"dripfeed://"
NOTE 1: The hostname part of a "file:" URL shall always be the empty string. These URLs are always in the namespace of the receiver. Transmitting them across a network as part of an application is meaningless.	
NOTE 2: DVB locators including the "dwb_abs_path" element may be returned by MHP APIs as a mechanism to provide references to files in carousels which may not currently be mounted. These locators can only be used to mount new carousels or be translated into a "file:" URL once a new carousel has been mounted.	

The DVB specifications define two places where multiple logical service components can be carried in a single MPEG elementary stream - DVB subtitles and MPEG-2 multichannel audio. This specification does not provide locators to distinguish between multiple languages of subtitles or audio carried in a single MPEG elementary stream in this way. Hence methods returning locators for service components / elementary streams shall return the same locator regardless of any selection between such logical service components. Methods accepting locators for service components / elementary streams shall select between any such logical service components based on the rules for elementary stream selection in section 11.4.2.2, "Clarifications" on page 250.

## 14.9 Service identification

In the MHP, there are two mechanisms for uniquely identifying a service:

- the triplet of numeric SI identifiers:

`original_network_id`, `transport_stream_id` and `service_id` (corresponding to identifiers with the same name defined by EN 300 468 [4] carried in the SI of the broadcast)

- a textual service identifier:

`textual_service_identifier_bytes` carried in the optional *Service identifier descriptor* in the SDT (see 10.12.1 on page 235).

Both enable global, unique identification of a service.

The textual identifier has additional properties:

- They can identify two (or more) service instances as being the same service even if they for technical reasons have different numeric identifiers.

It is up to the service provider to decide whether different service instances are identified as being the same service.

- They can give alternative identifications for a single service.

### 14.9.1 Syntax of the textual service identifier

The syntax of the textual service identifier is:

```
<service_name> "." <service_provider_domain_name>
```

where:

**<service\_name>**: is a unique name for the service within the service provider's domain

**<service\_provider\_domain\_name>**: is an Internet DNS domain name that the service provider has rights to control. The organization's administrating the Internet DNS domain names are used as a globally unique registration mechanism that allows these textual service identifiers to be globally unique names.

The <service\_name> field shall follow the rules defined for Internet DNS names so that the whole textual service identifier is a valid host name to be used in the Internet DNS as defined in RFC 1035 [75].

An example of a textual service identifier is:

```
movie_channel_1.broadcaster_b.com
```

where "broadcaster\_b.com" is an Internet DNS domain owned by the broadcaster and "movie\_channel\_1" is a unique name for the service assigned by the service provider

**NOTE:** The textual service identifier has the same syntax as an Internet host name and it must be assigned in a domain that the service provider has the rights to control. However, the textual service name for a service is not required to resolve to any IP address using the Internet DNS service and if it does, this version of this specification does not specify any specific services that this host should provide if contacted using the IP protocols.

## 14.9.2 Handling of the textual service identifiers within the MHP terminal

The MHP terminal discovers the textual service identifiers for a given service from the SDT table similarly as it discovers the existence of the service in the first place. The MHP terminal shall know the textual service identifiers for the available services in the same way that it knows the numeric identifiers: `original_network_id`, `transport_stream_id` and `service_id`.

When the application uses a URI referring to a DVB service, the resolution of this URI to the necessary information needed to locate the service happens in the same way regardless of if this URI contains a textual identifier or the numeric identifiers.

The URI string provided by the application shall be considered to match the one included in the SDT when the strings are the same.

## 15 Detailed platform profile definitions

This chapter defines the capabilities of platforms as presented to applications. Products that claim to conform to a profile shall provide at least the minimum capabilities identified for the profile. In some cases this implies that specific hardware resources are present in the platform.

**Table 136 : Detailed platform profile definitions (Sheet 1 of 3)**

Area	Specification	Enhanced Broadcast Profile 2	Interactive Broadcast Profile 2	Internet Access Profile 2
<b>Static formats</b>				
Bitmap pictures	7.1.1.3, "PNG" on page 64 + 15.1, "PNG - restrictions" on page 363	M	M	M
	7.1.1.3, "PNG" on page 64 without restrictions	n/r	n/r	n/r (note 3)
	7.1.1.4, "GIF" on page 64	n/r	n/r	n/r (note 3)
	7.1.2, "MPEG-2 I-Frames" on page 64	M	M	M
	7.1.1.2, "JPEG" on page 64 + 15.3, "JPEG - restrictions" on page 364	M	M	M
	7.1.1.2, "JPEG" on page 64	-	M	M
Audio clips	7.1.4, "Monomedia format for audio clips" on page 66	M	M	M
Video drips	7.1.3, "MPEG-2 Video "drips"" on page 64	M	M	M
Text encoding	7.1.5, "Monomedia format for text" on page 66	M	M	M
<b>Broadcast streaming formats</b>				
Video	7.2.2, "Video" on page 66	M	M	M
Audio	7.2.1, "Audio" on page 66	M	M	M
Subtitles	7.2.3, "Subtitles" on page 66	M	M	M
<b>Fonts</b>				
Built in	Character set see annex E, "(normative): Character set" on page 434, Metrics see annex D, "(normative): Text presentation" on page 421 Face: UK RNIB "Tiresias"	M	M	M
Downloadable	7.4, "Downloadable Fonts" on page 67	M	M	M
<b>Broadcast channel protocols</b>				
	6.2.2, "MPEG-2 Sections" on page 55	M	M	M
	6.2.5, "DSM-CC User-to-User Object Carousel" on page 55	M	M	M
	IP Multicast stack based on: 6.2.6, "DVB Multiprotocol Encapsulation" on page 56, 6.2.7, "Internet Protocol (IP)" on page 56 6.2.8, "User Datagram Protocol (UDP)" on page 56	O	Ro	M
<b>Interaction channel protocols</b>				
TCP/IP	6.3.3, "Transmission Control Protocol (TCP)" on page 57 6.3.2, "Internet Protocol (IP)" on page 57	-	M	M
UDP/IP	6.2.8, "User Datagram Protocol (UDP)" on page 56 6.3.2, "Internet Protocol (IP)" on page 57	-	M	M
DSM-CC U-U RPC	6.2.5, "DSM-CC User-to-User Object Carousel" on page 55 6.3.4, "UNO-RPC" on page 57 6.3.5, "UNO-CDR" on page 57	-	O	O
HTTP	6.3.7.1, "HTTP 1.1" on page 58	-	O	O
HTTP	6.3.7.2, "MHP profile of HTTP 1.0" on page 58	-	M	M
HTTPS	6.3.7.3, "HTTPS" on page 59	-	M	M

Table 136 : Detailed platform profile definitions (Sheet 2 of 3)

Area	Specification	Enhanced Broadcast Profile 2	Interactive Broadcast Profile 2	Internet Access Profile 2
Interaction Channel File System	6.4.1, "File system implemented only via the interaction channel" on page 60	-	M	M
DSMCC / HTTP hybrid	6.4.2, "Hybrid between broadcast stream and interaction channel" on page 62	-	M	M
<b>DVB-HTML</b>				
DVB-HTML	8, "DVB-HTML" on page 71	n/r	O (note 2)	O (note 2)
<b>DVB-J</b>				
Core	11.3, "Fundamental DVB-J APIs" on page 243	M	M	M
Presentation	11.4.1, "Graphical User Interface API" on page 246	M (note 1)	M (note 1)	M (note 1)
	11.4.2, "Streamed Media API" on page 250	M	M	M
Data Access	11.5.1, "Broadcast Transport Protocol Access API" on page 255	M	M	M
	11.5.2, "Support for Multicast IP over the Broadcast Channel" on page 257	O	Ro	M
	11.5.3, "Support for IP over the Return Channel" on page 258	-	M	M
	11.5.4, "MPEG-2 Section Filter API" on page 258	M	M	M
	11.5.5, "Mid-Level Communications API" on page 258	-	M	M
	11.5.6, "Persistent Storage API" on page 258	M	M	M
Service Information & Selection	11.6.1, "DVB Service Information API" on page 259	M	M	M
	11.6.2, "Service Selection API" on page 259	M	M	M
	11.6.3, "Tuning API" on page 259	M	M	M
	11.6.4, "Conditional Access API" on page 260	M	M	M
	11.6.5, "Protocol Independent SI API" on page 260	M	M	M
Common Infrastructure	11.7.1, "APIs to support DVB-J application lifecycle" on page 261	M	M	M
	11.7.2, "Application discovery and launching APIs" on page 261	M	M	M
	11.7.3, "Inter-Application and Inter-Xlet communication API" on page 263	M	M	M
	11.7.4, "Basic MPEG Concepts" on page 265	M	M	M
	11.7.5, "Resource Notification" on page 266	M	M	M
	11.7.6, "Content Referencing" on page 266	M	M	M
	11.7.7, "Common Error Reporting" on page 267	M	M	M
	11.7.8, "Plug-in APIs" on page 267	M	M	M
Security	11.8.1, "Basic Security" on page 268	M	M	M
	11.8.2, "APIs to Support TLS / SSL Over the Return Channel" on page 269	-	M	M
	11.8.3, "Additional permissions classes" on page 269	M	M	M
Others	11.9.1, "Timer Support" on page 269	M	M	M
	11.9.2, "User Settings and Preferences API" on page 270	M	M	M
	11.9.3, "Profile and version properties" on page 270	M	M	M
	11.9.4, "Non-CA smart card API" on page 271	M	M	M
Stand-alone applications	11.12.2, "Stored services" on page 281	M	M	M
Support for DVB-HTML	11.13, "Support for DVB-HTML" on page 282	O (note 2)	O (note 2)	O (note 2)

**Table 136 : Detailed platform profile definitions (Sheet 3 of 3)**

Area	Specification	Enhanced Broadcast Profile 2	Interactive Broadcast Profile 2	Internet Access Profile 2
Internet access	11.14, "Internet access" on page 283	-	-	M
	17, "Internet access clients" on page 369	-	-	M
NOTE 1: The javax.tv.graphics.TVContainer.getRootContainer method shall return an instance of org.havi.ui.HScene or null except when called by an embedded Xlet where the behaviour is defined in 11.13.2, "Embedded Xlets in a DVB-HTML Page" on page 282.				
NOTE 2: DVB-HTML is a single option which shall only be considered included if all parts of it are included.				
NOTE 3: Recommended to be implemented as part of WWW browser but outside the scope of the MHP conformance regime				

Where an MHP product claims to support a feature defined as optional, recommended optional or not required in the above table, it shall be supported according to this specification, shall be subject to the MHP conformance regime and shall report that the feature is supported using the mechanisms defined in table 118, "System properties for optional feature interrogation" on page 271.

Key	
-	Not applicable
n/r	Not required
O	Optional feature in the receiver
Ro	Recommended optional feature in the receiver
M	Mandatory feature in the receiver

## 15.1 PNG - restrictions

Engines are required to support ALL of the PNG colour types defines in PNG Specification Version 1.0 (see table 137). Engines are responsible for mapping these colours to those used by the engine's OSD.

Any combination of PNGs with different colour types may be active at any one time. Similarly, engines are responsible for mapping RGB16 direct colour specifications to colours that the OSD can support.

**Table 137 : PNG Formats**

Colour Type	Allowed Bit Depths	Interpretation
0	1, 2, 4, 8, 16	Each pixel is a grayscale sample.
2	8, 16	Each pixel is an R,G,B triple.
3	1, 2, 4, 8	Each pixel is a palette index; PLTE chunk must appear.
4	8, 16	Each pixel is a grayscale sample, followed by an alpha sample.
6	8, 16	Each pixel is an R,G,B triple, followed by an alpha sample.

Where PNG graphics use colours defined in the currently active application palette these colours shall be reproduced correctly. Other colours shall be reproduced in an implementation dependent way.

Receivers should ignore gAMA (gamma) and cHRM (chromaticity) chunks in PNG files.

In this profile, all PNG graphics should be gamma corrected.

### 15.1.1 PNG Aspect ratios

PNG bitmaps shall carry a pHYS chunk indicating the pixel aspect ratio of the bitmap. This aspect ratio should be the same as that of the scene containing the bitmap.

The PNG specification indicates that if the aspect ratio is absent square pixels should be assumed. To avoid overriding this specification the aspect ratio should be signalled explicitly.

## 15.2 Minimum media formats supported by DVB-J APIs

The following table specifies the minimum set of media types that implementations of the "Enhanced Broadcast Profile 1" and "Interactive Broadcast Profile 1" shall support. It also identifies the APIs that shall provide this support:

**Table 138 : Media type support required in Enhanced and Interactive Broadcast profile 1**

Media type	Section Reference	API(s)
Downloadable fonts	7.4	java.awt.Font
Audio from file	7.1.4	org.havi.ui.HSound JMF
MPEG I frame images	7.1.1	org.havi.ui.HBackgroundImage
PNG images	7.1.1.3	java.awt.Image
JPEG images	7.1.1.2	java.awt.Image
DVB service	7.2	JMF
Video "drips"	7.1.3	org.dvb.media.DripFeedDataSource

## 15.3 JPEG - restrictions

The restricted JPEG specification is as specified in 7.1.1.2, "JPEG" on page 64 except that the "progressive DCT-based" mode is excluded.

## 15.4 Locale support

Support of resources for the following locales is required:

- one guaranteed one (EN.UK)
- zero (or more) implementation dependant ones

Further it is guaranteed that the default Locale shall have resources. The default Locale is implementation dependant.



## 16 Registry of Constants

### 16.1 System constants

Table 139 : Registry of constants

Entity	Value	Description
PTimerMinRepeatInterval	40 ms	This (or optionally a smaller) value shall be returned by javax.tv.util.TVTimer.getMinRepeatInterval(). See 11.9.1, "Timer Support" on page 269.
PTimerGranularity	10ms	This (or optionally a smaller) value shall be returned by javax.tv.util.TVTimer.getGranularity(). See 11.9.1, "Timer Support" on page 269.

Table 140 : Profile encoding

application profile	version			Definition
	major	minor	micro	
1	1	1	0	Enhanced Broadcast Profile 2 as defined in this specification
2	1	1	0	Interactive Broadcast Profile 2 as defined in this specification
3	1	1	0	Internet Access Profile 2 as defined in this specification

### 16.2 DVB-J constants

This section to be populated with the values of public final static symbols from the various Java APIs.

#### 16.2.1 Public and Protected final static primitive fields from DVB packages

The following is a list of the values assigned for public and protected final static primitive fields defined in the DVB defined DVB-J packages:

```

public final static int org.dvb.application.AppAttributes.DVB_J_application = 1;
public final static int org.dvb.application.AppAttributes.DVB_HTML_application = 2;
public final static int org.dvb.application.AppProxy.STARTED = 0;
public final static int org.dvb.application.AppProxy.DESTROYED = 1;
public final static int org.dvb.application.AppProxy.NOT_LOADED = 2;
public final static int org.dvb.application.AppProxy.PAUSED = 3;
public final static int org.dvb.application.AppsDatabaseEvent.NEW_DATABASE = 0;
public final static int org.dvb.application.AppsDatabaseEvent.APP_CHANGED = 1;
public final static int org.dvb.application.AppsDatabaseEvent.APP_ADDED = 2;
public final static int org.dvb.application.AppsDatabaseEvent.APP_DELETED = 3;
public final static int org.dvb.application.DVBProxy.LOADED = 5;

public final static int org.dvb.dsmcc.DSMCCObject.FROM_CACHE = 1;
public final static int org.dvb.dsmcc.DSMCCObject.FROM_CACHE_OR_STREAM = 2;
public final static int org.dvb.dsmcc.DSMCCObject.FROM_STREAM_ONLY = 3;

public final static int org.dvb.event.UserEvent.UEF_KEY_EVENT = 1;

public final static int org.dvb.io.persistent.FileAttributes.PRIORITY_LOW = 1;
public final static int org.dvb.io.persistent.FileAttributes.PRIORITY_MEDIUM = 2;
public final static int org.dvb.io.persistent.FileAttributes.PRIORITY_HIGH = 3;

public final static int org.dvb.media.PresentationChangedEvent.STREAM_UNAVAILABLE = 0;
public final static int org.dvb.media.PresentationChangedEvent.CA_FAILURE = 1;
public final static int org.dvb.media.PresentationChangedEvent.CA_RETURNED = 2;
public final static int org.dvb.media.VideoFormatControl.ASPECT_RATIO_UNKNOWN = -1;
public final static int org.dvb.media.VideoFormatControl.AFD_NOT_PRESENT = -1;

```

```

public final static int org.dvb.media.VideoFormatControl.DFC_PROCESSING_UNKNOWN = -1;
public final static int org.dvb.media.VideoFormatControl.DFC_PROCESSING_NONE = 0;
public final static int org.dvb.media.VideoFormatControl.DFC_PROCESSING_FULL = 1;
public final static int org.dvb.media.VideoFormatControl.DAR_4_3 = 1;
public final static int org.dvb.media.VideoFormatControl.ASPECT_RATIO_4_3 = 2;
public final static int org.dvb.media.VideoFormatControl.AFD_16_9_TOP = 2;
public final static int org.dvb.media.VideoFormatControl.DFC_PROCESSING_LB_16_9 = 2;
public final static int org.dvb.media.VideoFormatControl.DAR_16_9 = 2;
public final static int org.dvb.media.VideoFormatControl.ASPECT_RATIO_16_9 = 3;
public final static int org.dvb.media.VideoFormatControl.AFD_14_9_TOP = 3;
public final static int org.dvb.media.VideoFormatControl.DFC_PROCESSING_LB_14_9 = 3;
public final static int org.dvb.media.VideoFormatControl.ASPECT_RATIO_2_21_1 = 4;
public final static int org.dvb.media.VideoFormatControl.AFD_GT_16_9 = 4;
public final static int org.dvb.media.VideoFormatControl.DFC_PROCESSING_CCO = 4;
public final static int org.dvb.media.VideoFormatControl.DFC_PROCESSING_PAN_SCAN = 5;
public final static int org.dvb.media.VideoFormatControl.DFC_PROCESSING_LB_2_21_1_ON_4_3 = 6;
public final static int org.dvb.media.VideoFormatControl.DFC_PROCESSING_LB_2_21_1_ON_16_9 = 7;
public final static int org.dvb.media.VideoFormatControl.AFD_SAME = 8;
public final static int org.dvb.media.VideoFormatControl.DFC_PLATFORM = 8;
public final static int org.dvb.media.VideoFormatControl.AFD_4_3 = 9;
public final static int org.dvb.media.VideoFormatControl.AFD_16_9 = 10;
public final static int org.dvb.media.VideoFormatControl.AFD_14_9 = 11;
public final static int org.dvb.media.VideoFormatControl.AFD_4_3_SP_14_9 = 13;
public final static int org.dvb.media.VideoFormatControl.AFD_16_9_SP_14_9 = 14;
public final static int org.dvb.media.VideoFormatControl.AFD_16_9_SP_4_3 = 15;
public final static byte org.dvb.media.VideoPresentationControl.POS_CAP_OTHER = -1;
public final static byte org.dvb.media.VideoPresentationControl.POS_CAP_FULL = 0;
public final static byte org.dvb.media.VideoPresentationControl.POS_CAP_FULL_IF_ENTIRE_VIDEO_ON_SCREEN=1;
public final static byte org.dvb.media.VideoPresentationControl.POS_CAP_FULL_EVEN_LINES = 3;
public final static byte org.dvb.media.VideoPresentationControl.POS_CAP_FULL_EVEN_LINES_IF_ENTIRE_VIDEO_ON_SCREEN = 4;

public final static int org.dvb.net.rc.RCInterface.TYPE_PSTN = 1;
public final static int org.dvb.net.rc.RCInterface.TYPE_ISDN = 2;
public final static int org.dvb.net.rc.RCInterface.TYPE_DECT = 3;
public final static int org.dvb.net.rc.RCInterface.TYPE_CATV = 4;
public final static int org.dvb.net.rc.RCInterface.TYPE_LMDS = 5;
public final static int org.dvb.net.rc.RCInterface.TYPE_MATV = 6;

public final static short org.dvb.si.DescriptorTag.NETWORK_NAME = 64;
public final static short org.dvb.si.DescriptorTag.SERVICE_LIST = 65;
public final static short org.dvb.si.DescriptorTag.STUFFING = 66;
public final static short org.dvb.si.DescriptorTag.SATELLITE_DELIVERY_SYSTEM = 67;
public final static short org.dvb.si.DescriptorTag.CABLE_DELIVERY_SYSTEM = 68;
public final static short org.dvb.si.DescriptorTag.BOUQUET_NAME = 71;
public final static short org.dvb.si.DescriptorTag.SERVICE = 72;
public final static short org.dvb.si.DescriptorTag.COUNTRY_AVAILABILITY = 73;
public final static short org.dvb.si.DescriptorTag.LINKAGE = 74;
public final static short org.dvb.si.DescriptorTag.NVOD_REFERENCE = 75;
public final static short org.dvb.si.DescriptorTag.TIME_SHIFTED_SERVICE = 76;
public final static short org.dvb.si.DescriptorTag.SHORT_EVENT = 77;
public final static short org.dvb.si.DescriptorTag.EXTENDED_EVENT = 78;
public final static short org.dvb.si.DescriptorTag.TIME_SHIFTED_EVENT = 79;
public final static short org.dvb.si.DescriptorTag.COMPONENT = 80;
public final static short org.dvb.si.DescriptorTag.MOSAIC = 81;
public final static short org.dvb.si.DescriptorTag.STREAM_IDENTIFIER = 82;
public final static short org.dvb.si.DescriptorTag.CA_IDENTIFIER = 83;
public final static short org.dvb.si.DescriptorTag.CONTENT = 84;
public final static short org.dvb.si.DescriptorTag.PARENTAL_RATING = 85;
public final static short org.dvb.si.DescriptorTag.TELETEXT = 86;
public final static short org.dvb.si.DescriptorTag.TELEPHONE = 87;
public final static short org.dvb.si.DescriptorTag.LOCAL_TIME_OFFSET = 88;
public final static short org.dvb.si.DescriptorTag.SUBTITLING = 89;
public final static short org.dvb.si.DescriptorTag.TERRESTRIAL_DELIVERY_SYSTEM = 90;
public final static short org.dvb.si.DescriptorTag.MULTILINGUAL_NETWORK_NAME = 91;
public final static short org.dvb.si.DescriptorTag.MULTILINGUAL_BOUQUET_NAME = 92;
public final static short org.dvb.si.DescriptorTag.MULTILINGUAL_SERVICE_NAME = 93;
public final static short org.dvb.si.DescriptorTag.MULTILINGUAL_COMPONENT = 94;
public final static short org.dvb.si.DescriptorTag.PRIVATE_DATA_SPECIFIER = 95;
public final static short org.dvb.si.DescriptorTag.SERVICE_MOVE = 96;
public final static short org.dvb.si.DescriptorTag.SHORT_SMOOTHING_BUFFER = 97;

```

```

public final static short org.dvb.si.DescriptorTag.FREQUENCY_LIST = 98;
public final static short org.dvb.si.DescriptorTag.PARTIAL_TRANSPORT_STREAM = 99;
public final static short org.dvb.si.DescriptorTag.DATA_BROADCAST = 100;
public final static byte org.dvb.si.PMTStreamType.MPEG1_VIDEO = 1;
public final static byte org.dvb.si.PMTStreamType.MPEG2_VIDEO = 2;
public final static byte org.dvb.si.PMTStreamType.MPEG1_AUDIO = 3;
public final static byte org.dvb.si.PMTStreamType.MPEG2_AUDIO = 4;
public final static short org.dvb.si.SIInformation.FROM_CACHE_ONLY = 0;
public final static short org.dvb.si.SIInformation.FROM_CACHE_OR_STREAM = 1;
public final static short org.dvb.si.SIInformation.FROM_STREAM_ONLY = 2;
public final static byte org.dvb.si.SIMonitoringType.NETWORK = 1;
public final static byte org.dvb.si.SIMonitoringType.BOUQUET = 2;
public final static byte org.dvb.si.SIMonitoringType.SERVICE = 3;
public final static byte org.dvb.si.SIMonitoringType.PMT_SERVICE = 4;
public final static byte org.dvb.si.SIMonitoringType.PRESENT_FOLLOWING_EVENT = 5;
public final static byte org.dvb.si.SIMonitoringType.SCHEDULED_EVENT = 6;
public final static byte org.dvb.si.SIRunningStatus.UNDEFINED = 0;
public final static byte org.dvb.si.SIRunningStatus.NOT_RUNNING = 1;
public final static byte org.dvb.si.SIRunningStatus.STARTS_IN_A_FEW_SECONDS = 2;
public final static byte org.dvb.si.SIRunningStatus.PAUSING = 3;
public final static byte org.dvb.si.SIRunningStatus.RUNNING = 4;
public final static short org.dvb.si.SIServiceType.UNKNOWN = -1;
public final static short org.dvb.si.SIServiceType.DIGITAL_TELEVISION = 1;
public final static short org.dvb.si.SIServiceType.DIGITAL_RADIO_SOUND = 2;
public final static short org.dvb.si.SIServiceType.TELETEXT = 3;
public final static short org.dvb.si.SIServiceType.NVOD_REFERENCE = 4;
public final static short org.dvb.si.SIServiceType.NVOD_TIME_SHIFTED = 5;
public final static short org.dvb.si.SIServiceType.MOSAIC = 6;
public final static short org.dvb.si.SIServiceType.PAL = 7;
public final static short org.dvb.si.SIServiceType.SECAM = 8;
public final static short org.dvb.si.SIServiceType.D_D2_MAC = 9;
public final static short org.dvb.si.SIServiceType.FM_RADIO = 10;
public final static short org.dvb.si.SIServiceType.NTSC = 11;
public final static short org.dvb.si.SIServiceType.DATA_BROADCAST = 12;
public final static short org.dvb.si.SIServiceType.MHP_APPLICATION = 16;

public final static int org.dvb.test.DVBTest.UNTESTED = -5;
public final static int org.dvb.test.DVBTest.UNRESOLVED = -4;
public final static int org.dvb.test.DVBTest.HUMAN_INTERVENTION = -3;
public final static int org.dvb.test.DVBTest.OPTION_UNSUPPORTED = -2;
public final static int org.dvb.test.DVBTest.FAIL = -1;
public final static int org.dvb.test.DVBTest.PASS = 0;

public final static int org.dvb.ui.DVBAlphaComposite.CLEAR = 1;
public final static int org.dvb.ui.DVBAlphaComposite.SRC = 2;
public final static int org.dvb.ui.DVBAlphaComposite.SRC_OVER = 3;
public final static int org.dvb.ui.DVBAlphaComposite.DST_OVER = 4;
public final static int org.dvb.ui.DVBAlphaComposite.SRC_IN = 5;
public final static int org.dvb.ui.DVBAlphaComposite.DST_IN = 6;
public final static int org.dvb.ui.DVBAlphaComposite.SRC_OUT = 7;
public final static int org.dvb.ui.DVBAlphaComposite.DST_OUT = 8;
public final static int org.dvb.ui.DVBBufferedImage.TYPE_ADVANCED = 20;
public final static int org.dvb.ui.DVBBufferedImage.TYPE_BASE = 21;
public final static int org.dvb.ui.DVBTextLayoutManager.HORIZONTAL_START_ALIGN = 1;
public final static int org.dvb.ui.DVBTextLayoutManager.HORIZONTAL_END_ALIGN = 2;
public final static int org.dvb.ui.DVBTextLayoutManager.HORIZONTAL_CENTER = 3;
public final static int org.dvb.ui.DVBTextLayoutManager.VERTICAL_START_ALIGN = 4;
public final static int org.dvb.ui.DVBTextLayoutManager.VERTICAL_END_ALIGN = 5;
public final static int org.dvb.ui.DVBTextLayoutManager.VERTICAL_CENTER = 6;
public final static int org.dvb.ui.DVBTextLayoutManager.LINE_ORIENTATION_HORIZONTAL = 10;
public final static int org.dvb.ui.DVBTextLayoutManager.LINE_ORIENTATION_VERTICAL = 11;
public final static int org.dvb.ui.DVBTextLayoutManager.START_CORNER_UPPER_LEFT = 20;
public final static int org.dvb.ui.DVBTextLayoutManager.START_CORNER_UPPER_RIGHT = 21;
public final static int org.dvb.ui.DVBTextLayoutManager.START_CORNER_LOWER_LEFT = 22;
public final static int org.dvb.ui.DVBTextLayoutManager.START_CORNER_LOWER_RIGHT = 23;

```

## 16.2.2 Public and Protected final static primitive fields from standard Java packages

These constants are recorded in the following documents:

- [JAE 1.1.8 const \[72\]](#)
- [JAE 1.2.2 const \[73\]](#)
- [JMF const \[74\]](#)

NOTE: The constants for [Java TV \[51\]](#) are within that specification. [JSSE \[60\]](#) introduces no constants.

## 17 Internet access clients

### 17.1 Referencing DVB services and content within WWW content

MHP internet access clients shall support the following mechanisms of referencing DVB services & content from within WWW content.

- Use of a reference to a DVB service in an HTML "href" shall launch that DVB service in the same way as is done by MHP applications using the MHP service selection API or the end user of the MHP terminal using the navigator to select DVB services.

MHP internet access clients which support a particular element.attribute from tables 12, "Use of ContentType attribute on elements" on page 83 and 14, "Use of MIME media type by element" on page 84 to a specific MIME type listed in the table shall support the DVB-HTML behaviour defined for that type of reference. This applies to the following MIME types only:

audio/mpeg, video/mpeg

multipart/dvb.service

NOTE: MHP applications shall only be started in response to selection of a reference to a DVB service. All MHP applications run in the scope of a DVB service and cannot run on their own outside of a service of some type.

### 17.2 Minimum requirements for internet clients

The following table defines some minimum network protocols which shall be supported by those internet client applications which form part of the MHP internet access profile.

Table 141 :

Internet Client Application	Minimum required protocols
WWW browser	HTTP (see 6.3.7.2, "MHP profile of HTTP 1.0" on page 58) HTTPS (see 6.3.7.3, "HTTPS" on page 59)
Email client	SMTP for sending email (see SMTP [93]) or HTTP to a WebMail server. Protocols for receiving email are implementation dependent.
Usenet news	NNTP (NNTP [94]) or HTTP to a WebNews server.

WWW browser internet clients shall support use of the "mailto" URL as defined in RFC 2368 [100].

Email internet clients shall support use of the "http" URL as defined in section 3.2.2 of RFC 2616 [40].

### 17.3 Internet streamed media

There is no requirement for MHP terminals to support internet streaming media content however where these are both supported and visible to DVB-J applications, it shall be done using the Java Media Framework (Java Media Player Specification [33]).

---

## Annex A (normative): External references; errata, clarifications and exemptions

This section lists known errata in normative external references, as well as clarifications to those references and/or exemptions from requirements in those specifications.

### A.1 JAE 1.1.8 API [31]

#### A.1.1 `java.lang.ThreadGroup.getParent()`

This specification is considered to include:

This method may throw a security exception.

#### A.1.2 `java.net.URLConnection.setFileNameMap`

The method `java.net.URLConnection.setFileNameMap` is considered to specify:

If there is a security manager, its `checkSetFactory` method is called with no arguments. This may result in a security exception.

#### A.1.3 `java.util.Locale.setDefault`

This specification is considered to include:

This method may throw a security exception.

#### A.1.4 `java.lang.Class`

The following clarifications are considered to be part of this specification:

Reflective operations, such as `Class.forName(String)` shall cause class initialization of the given class, if the call succeeds and the class had not previously been initialized.

#### A.1.5 `java.awt.Font`

Where an MHP terminal does not have available a font matching the specification of the parameters passed to the constructor of the `java.awt.Font` class, the implementation shall silently approximate to a font which is available. No runtime error or exception shall be thrown by the constructor. The approximation shall not be visible to implementations through any of the methods on the returned instance of `java.awt.Font`.

### A.2 Java Language Spec [32]

#### A.2.1 `java.lang.ThreadGroup.getParent()`

This method may throw a security exception.

#### A.2.2 `java.lang.Runtime.runFinalizersOnExit()`

This method may throw a `SecurityException`

#### A.2.3 `java.lang.System.runFinalizersOnExit()`

This method may throw a `SecurityException`

## A.3 Java Media Player Specification [33]

### A.3.1 javax.media.protocol.URLDataSource.sources

The `javax.media.protocol.URLDataSource.sources` field shall be considered not to be present. So, any reference to it will fail.

## A.4 Java VM [34]

The following clarifications are considered to be part of this specification:

- As described in sections 12.1.2 and 12.3.1 of *Java Language Spec* [32], class file verification is mandatory for all implementations that conform to *Java VM* [34].
- Resolving a class causes its superinterfaces to be resolved.
- As specified in section 13.1 of *Java Language Spec* [32], the target of the `invokeinterface` instruction's method invocation must support the referenced interface. If it does not, an `IncompatibleClassChangeError` shall be raised when the `invokeinterface` instruction is executed.

NOTE: This is described on page 282 of *Java VM2* [F].

Additionally for MHP terminals:

- MHP terminals are exempt from implementing class finalization, as required by section 2.16.8 of *Java VM* [34].

NOTE: As noted in the appendix of *Java VM2* [F], class finalization has not been implemented, and it is not required in the 2nd edition of the VM specification.

## A.5 Java TV [51]

The following is considered to be present:

All methods in *JavaTV* whose name is of the form `removeXXXXListener`, shall have no effect if the listener concerned is not registered.

### A.5.1 javax.tv.service.selection

#### A.5.1.1 PresentationTerminatedEvent

The following specification additions are considered to be present:

When a call to `ServiceContext.select()` fails for a service context in the not presenting state, the following table defines how the reason code for the `PresentationTerminatedEvent` shall be derived from the reason code of the `SelectionFailedEvent` which first notified applications of the failure of the method call.

**Table A.1 : Reason code mapping**

SelectionFailedEvent reason code	PresentationTerminatedEvent reason code
CA_REFUSAL	ACCESS_WITHDRAWN
CONTENT_NOT_FOUND	SERVICE_VANISHED
INSUFFICIENT_RESOURCES	RESOURCES_REMOVED
MISSING_HANDLER	RESOURCES_REMOVED
TUNING_FAILURE	TUNED_AWAY

No equivalent of the reason code `SelectionFailedEvent.INTERRUPTED` shall be generated for service contexts formerly in the `not_presenting` state since by definition, another selection is in process on the service context concerned.

In the definition of the `USER_STOP` code, replace "The user" with "An application or the end user".

### A.5.1.2 ServiceContext.select( Locator [] )

- a) The following text shall be considered as being added to the end of the description of this method.

If the content corresponding to any of the locators specified can be successfully presented then the selection operation shall be considered to have succeeded even if attempts to present content corresponding to other locators failed. If different locators failed for different reasons then the reason code is implementation dependent.

- b) Replace the following text:

Successful completion of a select operation using this method provides `ServiceContentHandler` instances for all components that are indicated in the `components` parameter.

with

Successful completion of a selection operation using this method provides `ServiceContentHandlers` for those components indicated in the `components` parameter which were successfully presented.

## A.5.2 javax.tv.util.TVTimer

### A.5.2.1 scheduleTimerSpec(TVTimerSpec)

The following specification changes are considered to be present:

Replace:

If you schedule an absolute specification that should have gone off already, it will go off immediately.

With:

If you schedule an absolute specification that should have gone off already, it will go off immediately, and the return value of this method will be an absolute specification reflecting the current absolute time.

## A.5.3 javax.tv.util.TVTimerSpec

### A.5.3.1 setAbsoluteTime(long)

The following specification additions are considered to be present:

If the time parameter passed is negative, this method shall throw an `IllegalArgumentException`.

### A.5.3.2 setTime(long)

The following specification additions are considered to be present:

If the time parameter passed is negative, this method shall throw an `IllegalArgumentException`.

## A.5.4 javax.tv.xlet.Xlet

### A.5.4.1 Xlet state descriptions

In the table of xlet state descriptions in the package description of the `javax.tv.xlet` package, the description of the loaded state is considered to be modified as follows.

Replace "created using new" with "created using `Class.newInstance`".

### A.5.4.2 Own service context

The specification is considered to include

An Xlet's "own" `ServiceContext` instance is one where the execution of that Xlet is part of a currently selected service which can be controlled by that `ServiceContext` instance.



## **A.6**     **DAVIC 1.4.1p9 [3]**

### A.6.1     org.davic.mpeg.NotAuthorizedException

The specification is considered to include `org.davic.mpeg.NotAuthorizedException` as specified below:

# org.davic.mpeg NotAuthorizedException

## Syntax

```
public class NotAuthorizedException extends java.lang.Exception implements org.davic.mpeg.
    NotAuthorizedInterface
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--org.davic.mpeg.NotAuthorizedException
```

## All Implemented Interfaces:

NotAuthorizedInterface, java.io.Serializable

## Description

This class is thrown by MPEG related APIs when access is requested to information which is scrambled and to which access is not permitted by the security system.

## Constructors

### NotAuthorizedException()

```
public NotAuthorizedException()
```

Constructs a NotAuthorizedException with no detail message

### NotAuthorizedException(String)

```
public NotAuthorizedException(java.lang.String s)
```

Constructs a NotAuthorizedException with the specified detail message

#### Parameters:

*s* - the detail message

## Methods

### getElementaryStreams()

```
public ElementaryStream[] getElementaryStreams()
```

If `getType()` returns `ELEMENTARY_STREAM`, then this method returns the set of ElementaryStreams that could not be descrambled. Otherwise it returns null.

#### Specified By:

`NotAuthorizedInterface.getElementaryStreams()` in interface `NotAuthorizedInterface`

#### Returns:

either the set of ElementaryStreams that could not be descrambled or null

**getReason(int)**

```
public int[] getReason(int index)
```

Returns the reason(s) why descrambling was not possible.

**Specified By:**

`NotAuthorizedInterface.getReason(int)` in interface `NotAuthorizedInterface`

**Parameters:**

`index` - If the component to which access failed is a Service, `index` shall be 0. Otherwise `index` shall refer to one stream in the set returned by `getElementaryStreams()`.

**Returns:**

an array of length 2 where the first element of the array is the major reason and the second element of the array is the minor reason.

**Throws:**

`IndexOutOfBoundsException` - If the component to which access failed is a Service, this exception will be thrown if `index` is non zero. If the component(s) to which access failed was a (set of) elementary streams then this exception will be thrown where `index` is beyond the size of the array returned by `getElementaryStreams`.

**See Also:**

`NotAuthorizedInterface.getElementaryStreams()`

**getService()**

```
public Service getService()
```

If `getType()` returns `SERVICE`, then this method returns the Service that could not be descrambled. Otherwise it returns null.

**Specified By:**

`NotAuthorizedInterface.getService()` in interface `NotAuthorizedInterface`

**Returns:**

either the Service that could not be descrambled or null

**getType()**

```
public int getType()
```

**Specified By:**

`NotAuthorizedInterface.getType()` in interface `NotAuthorizedInterface`

**Returns:**

`SERVICE` or `ELEMENTARY_STREAM` to indicate that either a service (MPEG program) or one or more elementary streams could not be descrambled.

## A.6.2 org.davic.mpeg.dvb

### A.6.2.1 DvbElementaryStream

The following classes are considered to have a no argument protected constructor:

- DvbService
- DvbElementaryStream
- DvbTransportStream
- ElementaryStream
- Service
- TransportStream

## A.6.3 org.davic.mpeg.sections

### A.6.3.1 RingSectionFilter

Is considered to have the following text appended to its description:

All sections in a ring section filter are initialised to empty when the ring section filter is first created. Clearing them to empty any time after this is the responsibility of the application. Starting a ring section filter shall not clear any of the sections to empty.

### A.6.3.2 Section

#### A.6.3.2.1 clone()

Section is considered to have the method clone() with the following behaviour.

A cloned Section object is a new and separate object. It is unaffected by changes in the state of the original Section object or restarting of the SectionFilter the source Section object originated from. The clone method must be implemented without declaring exceptions.

#### A.6.3.2.2 Started Section Filters

The class description is considered to have the following text added at its end.

When a SectionFilterGroup is detached, either by the client or through resource withdrawal, started SectionFilters shall remain started. Hence if the SectionFilterGroup is re-attached, those filters shall re-activate.

#### A.6.3.2.3 getData()

Remove the following text from the methods of org.davic.mpeg.sections.Section:

(everything after the length field, not including a CRC check)

#### A.6.3.2.4 getFullStatus()

Is considered to have the following text appended to its description:

Returns true when the Section object contains valid data.

### A.6.3.3 SectionFilter

#### A.6.3.3.1 Cross reference error

In the description of this class there are 12 instances of a cross reference to H7. In each case this shall be considered as a reference to E.8.1.

#### A.6.3.3.2 `startFiltering(java.lang.Object, int, int)`

Is considered to have the following text appended to its description:

Note that this applied whether the parent section filter group is connected to a TS or not.

#### A.6.3.3.3 `startFiltering(java.lang.Object, int, int, int, byte[], byte[])`

Is considered to have the following text appended to its description:

`IllegalFilterDefinitionException` is thrown if offset is too small.

#### A.6.3.3.4 `startFiltering (appData, pid, tableId) exceptions`

Like other `startFiltering` methods `org.davic.mpeg.sections.SectionFilter.startFiltering (appData, pid, tableId)` shall throw an `IllegalFilterDefinitionException` where:

- the Java integer is negative
- the Java integer is larger than what is allowed for PID or `table_id` according to the relevant MPEG specification

### A.6.3.4 `SectionFilterGroup`

#### A.6.3.4.1 `NotAuthorizedException`

A `NotAuthorizedException` is added to the definition of this class.

#### A.6.3.4.2 `numberOfFilters`

Is considered to have the following text appended to its description:

Throws illegal argument exception if `numberOfFilters < 1`.

#### A.6.3.4.3 `sectionSize`

Is considered to have the following text appended to its description:

Throws illegal argument exception if `sectionSize < 1`.

#### A.6.3.4.4 `ringSize`

Is considered to have the following text appended to its description:

Throws illegal argument exception if `ringSize < 1`.

### A.6.3.5 `TimeoutEvent`

The specification is considered to include `org.davic.mpeg.sections.TimeoutEvent` as specified below:

# org.davic.mpeg.sections.TimeoutEvent

## Methods

### `getSource()`

```
public java.lang.Object getSource()
```

This returns the `SectionFilter` object which timed out

**Overrides:**

EndOfFilteringEvent.getSource() in class EndOfFilteringEvent

**A.6.4 org.davic.media****A.6.4.1 FreezeControl.resume()**

Add the following to the description of the semantics for this method:

If the player is started and if decoding of the media stream is not frozen then calls to this method shall have no effect otherwise an exception is thrown.

**A.6.4.2 MediaTimePositionChangedEvent**

Add the following constructor:

```
MediaTimePositionChangedEvent (
    Controller from,
    int previous,
    int current,
    int target,
    Time mediaTime)
```

**A.6.4.3 NotAuthorizedMediaException**

The following constructors are considered to be removed from the specification:

```
NotAuthorizedMediaException()
NotAuthorizedMediaException(java.lang.String reason)
```

The following constructors are considered to be a normative part of the specification:

```
NotAuthorizedMediaException( org.davic.mpeg.Service, int reason )
NotAuthorizedMediaException( org.davic.mpeg.ElementaryStream[], int reason[] )
```

**NotAuthorizedMediaException(ElementaryStream[], int[])**

```
public NotAuthorizedMediaException(org.davic.mpeg.ElementaryStream[] e, int[] reason)
```

Constructor for exception due to failure accessing one or more MPEG elementary streams The caller of this constructor is responsible for ensuring the two arrays provided as parameters are the same size. The implementation is not expected to check this.

**Parameters:**

e - the elementary streams which could not be accessed

reason - the reason why the exception was thrown for each elementary stream

**NotAuthorizedMediaException(Service, int)**

```
public NotAuthorizedMediaException(org.davic.mpeg.Service s, int reason)
```

Constructor for exception due to failure accessing an MPEG service

**Parameters:**

s - the service which could not be accessed

reason - the reason why the service could not be accessed

## A.6.5 org.davic.net

### A.6.5.1 InvalidLocatorException

The following class definition is considered to be a normative part of the specification:

# org.davic.net InvalidLocatorException

## Syntax

```
public class InvalidLocatorException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
|
+--java.lang.Exception
|
+--org.davic.net.InvalidLocatorException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This exception is thrown when one or more parameters to construct a Locator are invalid.

## Constructors

### InvalidLocatorException()

```
public InvalidLocatorException()
```

### InvalidLocatorException(String)

```
public InvalidLocatorException(java.lang.String reason)
```

Constructor for the exception with a specified reason

#### Parameters:

`reason` - the reason why the exception was raised

## A.6.5.2 Locator

### A.6.5.2.1 Locator()

The no-argument constructor for `org.davic.net.Locator` is considered to not be present. The absence of any description on the method indicates that this was an editing error in the DAVIC specification.

### A.6.5.2.2 toExternalForm()

Is considered to have the following text appended to its description:

If the instance of `Locator` has been created using `Locator(java.lang.String url)` and the URL is a non-null invalid URL the behaviour is implementation dependent.

### A.6.5.3 tuning

#### A.6.5.3.1 NetworkInterfaceController

##### A.6.5.3.1.1 reserve()

The semantic of the following throws clause is corrected as follows:

```
Throws: NoFreeInterfaceException
        raised if the requested network interface can not be reserved
```

The following from the semantic of this method:

If this NetworkInterfaceController has already reserved another NetworkInterface, then it will either release that NetworkInterface and reserve the specified one, or throw an exception. If the specified NetworkInterface has already been reserved by this NetworkInterfaceController, then this method does nothing.

is replaced with the following:

If this NetworkInterfaceController has currently reserved another NetworkInterface, then it will either release that NetworkInterface and reserve an appropriate one, or throw an exception. If a NetworkInterface that is able to tune to the specified transport stream is currently reserved by this NetworkInterfaceController, then this method does nothing.

##### A.6.5.3.1.2 reserveFor()

The following from the semantic of this method:

If this NetworkInterfaceController has already reserved another NetworkInterface, then it will either release that NetworkInterface and reserve an appropriate one, or throw an exception. If NetworkInterfaceController has already reserved a NetworkInterface that is able to tune to the specified transport stream, then this method does nothing.

is replaced with the following:

If this NetworkInterfaceController has currently reserved another NetworkInterface, then it will either release that NetworkInterface and reserve an appropriate one, or throw an exception. If a NetworkInterface that is able to tune to the specified transport stream is currently reserved by this NetworkInterfaceController, then this method does nothing.

##### A.6.5.3.1.3 tune()

Replace "this NetworkInterface" with "the NetworkInterface reserved by this NetworkInterfaceController" in section H.5.4.3 of DAVIC specification.

### A.6.5.4 ca

#### A.6.5.4.1 CAMessage

The following class definition is considered to be a normative part of the specification:

This class represents messages to ca modules.

```
public class CAMessage( byte[] data)
```

Constructor for the message

Parameters:

data - message data

Method Detail

getData

```
public byte[] getData()
```

Returns:



the data of the message

#### A.6.5.4.2 CAModule

##### A.6.5.4.2.1 buyEntitlement(org.davic.net.Locator)

This method is considered to have the amendment.

Replace:

Initiates a purchase dialogue for specified service or future event (specified by a Locator).

With:

Request to buy a specified service or future event (specified by a Locator) from a conditional access system.

In the comments of `org.davic.net.ca.CAModule.buyEntitlement()` the sentence:

In case of CA0 this maps onto `event_query` with `event_cmd_id = mmi` (Common Interface specification, section B.4.1.1)."

is replaced with

In case of DVB Common Interface, this maps onto CI messages as follows:

- when the Locator points to a service and the terminal is currently receiving the transport stream that this service is carried in and this transport stream is available to this CA module, then this method is mapped to a `ca_pmt` message with `ca_pmt_cmd_id` set to "ok\_mmi". The value returned in the `ca_pmt_reply` is mapped as defined in the documentation of the constants in the class. If the module is currently descrambling the service and the terminal is aware of this, `ENTITLEMENT_AVAILABLE` shall be returned immediately without communicating with the module.
- when the Locator points to a service that is not carried in a currently received transport stream, `NotTunedException` is thrown
- when the Locator points to an event, this maps onto `event_query` message with `event_cmd_id` set to "mmi" (Common Interface specification, section B.4.1.1). The value returned in the `event_reply` message is mapped as defined in the documentation of the constants in this class.

In the CA API, the constants defined in the `org.davic.net.ca.CAModule` class are mapped to the `CA_enable` values of the `ca_pmt_response` message and the `event_status` values of the `event_reply` message in the Common Interface protocol as follows:

**ENTITLEMENT\_AVAILABLE:**

CA\_enable value "Descrambling possible" (0x01)

event\_status value "entitlement\_available" (0x01)

event\_status value "mmi\_complete\_available" (0x05)

**ENTITLEMENT\_NOT\_AVAILABLE:**

CA\_enable value "Descrambling not possible (because no entitlement)" (0x04)

event\_status value "entitlement\_not\_available" (0x02)

event\_status value "mmi\_complete\_not\_available" (0x06)

**ENTITLEMENT\_UNKNOWN:**

CA\_enable value "Descrambling not possible (for technical reasons)" (0x05)

all other CA\_enable values not having an explicit mapping in this section

event\_status value "entitlement\_unknown" (0x00)

event\_status value "mmi\_complete\_unknown" (0x04)

all other event\_status values not having an explicit mapping in this section

**MMI\_DIALOGUE\_REQUIRED:**

CA\_enable value "Descrambling possible under conditions (purchase dialogue)" (0x02)

CA\_enable value "Descrambling possible under conditions (technical dialogue)" (0x03)

event\_status value "mmi\_dialogue\_required" (0x03)

#### A.6.5.4.2.2 isDescramblable(ElementaryStream streams[])

Is considered to have the following text appended to its description:

If an empty array is passed in, returns true.

#### A.6.5.4.2.3 openMessageSession(MessageListener)

This method is considered to have the amendment:

Modify:

Throws: ModuleBusyException

raised if the module is busy and is not able to handle a message session at the moment

To say:

Throws: ModuleBusyException

raised if the module is busy and is not able to handle a message session at the moment. This is CA system dependant.

#### A.6.5.4.2.4 queryEntitlement(org.davic.net.Locator)

This method is considered to include the following:

Throws: org.davic.net.InvalidLocatorException

if the locator does not point to a valid service or event

In the comments of `org.davic.net.ca.CAModule.queryEntitlement()` the sentence:

In case of CA0 this maps onto event\_query with event\_cmd\_id = query (Common Interface specification, section B.4.1.1)."

is replaced with

In case of DVB Common Interface, this maps onto CI messages as follows:

- when the Locator points to a service and the terminal is currently receiving the transport stream that this service is carried in and this transport stream is available to this CA module, then this method is mapped to a ca\_pmt message with ca\_pmt\_cmd\_id set to "query". The value returned in the ca\_pmt\_reply is mapped as defined in the documentation of the constants in the class. If the module is currently descrambling the service and the terminal is aware of this, ENTITLEMENT\_AVAILABLE shall be returned immediately without communicating with the module.
- when the Locator points to a service that is not carried in a currently received transport stream, ENTITLEMENT\_UNKNOWN shall be returned.
- when the Locator points to an event, this maps onto event\_query with event\_cmd\_id = query (Common Interface specification, section B.4.1.1). The return values is mapped as defined in the documentation of the constants in this class.

#### A.6.5.4.2.5 sendToModule

The description of this method is considered to have the following text added to it.

In systems based on the DVB common interface, messages sent using this method shall be mapped onto the CAPipelineRequest as defined in section 6.8.3 of the common interface extensions specification. Responses from the CA system reported through the CAPipelineResponse and CAPipelineNotification messages shall be mapped onto instances of ModuleResponseEvent.

### A.6.5.4.3 CAModuleManager

#### A.6.5.4.3.1 addMMIListener()

Is considered to have the following text appended to its description:

If an application has registered (and not removed) a listener to handle the MMI dialogues and if an MMI dialogue is required, this causes the platform to ask the MMI listener to handle the MMI dialogues. If there is no application registered to handle the MMI dialogues, these will be handled by the platform.

#### A.6.5.4.3.2 getModules(Service s)

Is considered to have the following text appended to its description:

If the service passed as a parameter is not scrambled, returns an empty array whose length is 0.

### A.6.5.4.4 NoFreeCapacityException

The following class definition is considered to be a normative part of the specification:

## org.davic.net.ca NoFreeCapacityException

### Syntax

```
public class NoFreeCapacityException extends org.davic.net.ca.CAException
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--org.davic.net.ca.CAException
            |
            +--org.davic.net.ca.NoFreeCapacityException
```

### All Implemented Interfaces:

java.io.Serializable

### Description

This exception is thrown when a method is called and the CA module does not have the required capacity to perform the action

## Constructors

### NoFreeCapacityException()

```
public NoFreeCapacityException()
```

Default constructor for the exception

### NoFreeCapacityException(String)

```
public NoFreeCapacityException(java.lang.String reason)
```

Constructor for the exception with a specified reason

**Parameters:**

`reason` - the reason why the exception was raised

A.6.5.4.5      **MMIObject**

The following class definition is considered to be a normative part of the specification:

# org.davic.net.ca MMIObject

**Syntax**

```
public class MMIObject
    java.lang.Object
    |
    +--org.davic.net.ca.MMIObject
```

**Direct Known Subclasses:****List, Text****Description**

The base class of all MMI classes.

**Methods****close()**

```
public void close()
```

Closes the MMI object and informs the CA API implementation that the application intends to close or has closed the corresponding MMI screen.

A.6.5.4.6      **DescramblerProxy**A.6.5.4.6.1      **startDescrambling()**

In the comments of `org.davic.net.ca.DescramblerProxy.startDescrambling()` (all signature versions) the sentence:

This method may start an MMI dialog.

is replaced with

This method may result in the CA system requesting an MMI dialog.

A.6.5.4.6.2      **startDescrambling(org.davic.mpeg.Service, java.lang.Object)**

Is considered to have the following text appended to its description:

DescramblerProxy applies from the point of view of one application. Methods such as `startDescrambling()` and `stopDescrambling` apply on a per-application basis and do not impact descrambling on behalf of other applications, except subject to platform resource limitations.

A.6.5.4.6.3 `startDescrambling(org.davic.mpeg.ElementaryStream[], java.lang.Object)`

Is considered to have the following text appended to its description:

If `org.davic.mpeg.ElementaryStream[]` is a zero length array the method has no effect.

A.6.5.4.6.4 `startDescrambling(org.davic.mpeg.ElementaryStream[], CAModule, java.lang.Object)`

Is considered to have the following text appended to its description:

If `org.davic.mpeg.ElementaryStream[]` is a zero length array the method has no effect.

A.6.5.4.6.5 `startDescramblingDialog(org.davic.mpeg.ElementaryStream[])`

Is considered to have the following text appended to its description:

If `org.davic.mpeg.ElementaryStream[]` is a zero length array the method has no effect.

A.6.5.4.6.6 `stopDescrambling()`

Is considered to have the following text appended to its description:

If no descrambling is being done then this method has no effect

A.6.5.4.6.7 `stopDescrambling(org.davic.mpeg.ElementaryStream[] streams)`

Is considered to have the following text appended to its description:

The method `stopDescrambling( ElementaryStream[] )` only stops the descrambling of streams which have been started through this `DescramblerProxy` instance, and not started through any other instance.

Is considered to include the following parameter specification:

**streams:** array of `ElementaryStreams` whose descrambling is to be stopped.

Is considered to have the following text appended to its description:

The `stopDescrambling` method only effects members of the array of streams that are being descrambled. There is no effect on any streams listed in the array that are not being descrambled.

A.6.5.4.7 `StartMMIEvent(MMIObject, int, java.lang.Object)`

Is considered to include the following parameter specification:

**caModule:** the `CAModule` object that is the source of the event, which shall be returned by the `getSource()` method.

A.6.5.5 `dvb.DvbLocator(int onid, int tsid, int serviceid, int eventid, int componenttags[], String filePath).`

The following parameter specification:

**the:** file path string including the slash character in the beginning

is considered to read:

**filePath:** string including the slash character in the beginning

[2332] . . Add

A.6.5.6 `dvb.DvbLocator`

The following method is considered to be present:

```

/** Returns the textual service identifier, if present.
 * @return the textual service identifier, null if not present
 * @since MHP1.0.1
 */
public String getTextualServiceIdentifier() ;

```

The returns clause of the `getOriginalNetworkId()` method is considered to include "-1 if not present" similarly as the other methods returning the numeric identifiers.

## A.7 CSS 2 [87]

This specification is considered to include:

### 11.1.2 Clipping: the "clip" property

A clipping region defines what portion of an element's rendered content is visible. By default, the clipping region has the same size and shape as the element's box(es). However, the clipping region may be modified by the "clip" property.

"clip"

Value: <shape> | auto | inherit

Initial: auto

Applies to: block-level and replaced elements

Inherited: no

Percentages: N/A

Media: visual

The "clip" property applies to elements that have a "overflow" property with a value other than "visible". Values have the following meanings:

auto

The clipping region has the same size and location as the element's box(es).

<shape>

In CSS2, the only valid <shape> value is: `rect (<top> <right> <bottom> <left>)` where <top>, <bottom>, <right>, and <left> specify offsets from the respective sides of the box.

<top>, <right>, <bottom>, and <left> may either have a <length> value or "auto". Negative lengths are permitted. The value "auto" means that a given edge of the clipping region will be the same as the edge of the element's generated box (i. e., "auto" means the same as "0").

When coordinates are rounded to pixel coordinates, care should be taken that no pixels remain visible when <left> + <right> is equal to the element's width (or <top> + <bottom> equals the element's height), and conversely that no pixels remain hidden when these values are 0.

The element's ancestors may also have clipping regions (in case their "overflow" property is not "visible"); what is rendered is the intersection of the various clipping regions.

If the clipping region exceeds the bounds of the UA's document window, content may be clipped to that window by the native operating environment.

Example(s):

The following two rules:

```
P { clip: rect(5px, 10px, 10px, 5px); }
```

```
P { clip: rect(5px, -5px, 10px, 5px); }
```

will create the rectangular clipping regions delimited by the dashed lines in the following illustrations:

[D]

Note. In CSS2, all clipping regions are rectangular. We anticipate future extensions to permit non-rectangular clipping.

While CSS2 specifies that values of "rect()" specify offsets from the respective sides of the box, current implementations interpret values with respect to the top and left edges for all four values (top, right, bottom, and left). The Working Group proposes to revise CSS2 to conform to current practice.

# Annex B (normative): Object carousel

## B.1 Introduction

The broadcast applications are transmitted using the DSM-CC User-to-User Object Carousels.

This specification is based on the following specifications:

- ISO/IEC 13818-1 [23] - MPEG 2 systems
- ISO/IEC 13818-6 [26] - DSM-CC
- EN 301 192 [5] - DVB specification for data broadcasting
- TR 101 202 [49] - Implementation Guidelines for Databroadcasting

With the constraints and extensions described here.

### B.1.1 Key to notation

Certain notations are used in the "value" columns of the syntax tables:

**Table B.1 : Key to notation**

Symbol	
+	A value that is "allocated" e.g. configuration parameter of the object carousel server.
*	A value that is "calculated" e.g. a field whose value is calculated by the carousel server as a consequence of the number of bytes in other fields

## B.2 Object Carousel Profile

In the following chapter, the message structures of the Object carousels are introduced with associated additional restrictions. Each section contains a table specifying the restrictions on the usage of the fields. The table also indicates the source for these restrictions: the DSM-CC standard, DVB guidelines or a specific restriction for this specification.

For the object carousel messages, also the message syntax is included. **In the syntax tables grey shading indicates parts that the broadcaster may put in, but an MHP terminal compliant with this specification may ignore.**

### B.2.1 DSM-CC Sections

All object carousels messages are transmitted using DSM-CC section format. The DSM-CC Section format is defined in chapter 9.2 of the DSM-CC specification.

The DSM-CC standard provides an option to use either a CRC32 or a checksum for detecting bit errors. For this specification, we make the following restriction:

**Table B.2 : Restrictions on DSM-CC Section format**

Field	Restrictions	Source
section_syntax_indicator	1 (indicating the use of the CRC32)	This spec.
last_section_number	For sections transporting DownloadDataBlock fragments: - all modules intended to be retrieved shall have the last section number $\leq 0xFE$ - if last section number = $0xFF$ implementations conforming to this spec are allowed to abort the retrieval and report error condition	This spec.

The maximum section length is 4096 bytes for all types of sections used in Object Carousels. The section overhead is 12 bytes, leaving a maximum of 4084 bytes of payload per section.



### B.2.1.1 Sections per TS packet

Any single TS packet is allowed to contain no more than the payload of two sections (i.e. the end of one section and the beginning of another).

## B.2.2 Data Carousel

This section defines the content of the data carousel messages when used in the object carousel.

### B.2.2.1 General

The definitions in [Table B.3](#) apply to both the `dsmccDownloadDataHeader` and the similar `dsmccMessageHeader`.

**Table B.3 : Restrictions on DSM-CC DownloadData and Message headers**

Field	Restrictions	Source
TransactionId	See "Assignment and use of transactionId values" on page 409	This spec.
AdaptationLength	The MHP terminal may ignore the possible contents of the <code>dsmccAdaptationHeader</code> field.	This spec.

### B.2.2.2 DownloadInfoIndication

The `DownloadInfoIndication` is a message that describes a set of modules and gives the necessary parameters to locate the module and retrieve it.

**Table B.4 : Restrictions on the DII**

Field	Restrictions	Source
blockSize	maximum size 4066 (max. section payload - DDB-header size (18)) The recommended blockSize is 4066.	DSM-CC (This spec. rec.)
windowSize	0 (not used for Object Carousels)	DSM-CC
ackPeriod	0 (not used for Object Carousels)	DSM-CC
tCDownloadWindow	0 (not used for Object Carousels)	DSM-CC
tCDownloadScenario	0 (not used for Object Carousels)	DSM-CC
compatibilityDescriptor(): compatibilityDescriptorLength	0 (no compatibility descriptor for Object Carousels)	DSM-CC
PrivateDataLength	The MHP terminal may ignore the possible contents of the <code>privateData</code> field	DVB

### B.2.2.3 DownloadServerInitiate

The `DownloadServerInitiate` is used in the case of object carousels to provide the object reference to the `ServiceGateway` (i.e. root directory) of the object carousel.

**Table B.5 : Restrictions on DSI**

Field	Restrictions	Source
compatibilityDescriptor(): compatibilityDescriptorLength	0 (no compatibility descriptor for Object Carousels)	DSM-CC
privateData	Contains the <code>ServiceGatewayInfo</code> structure	DSM-CC
serverId	Shall be set to 20 bytes each with the value of 0xFF	DVB / This spec.

### B.2.2.4 ModuleInfo

The moduleInfo structure is placed in the moduleInfo field of the DownloadInfoIndication of the data carousel. It contains the information needed to locate the module.

**Table B.6 : Restrictions on the DII moduleInfo field**

Field	Restrictions	Source
BIOP::ModuleInfo::Taps	The first tap shall have the "use" value 0x0017 (BIOP_OBJECT_USE). The id and selector fields are not used and the MHP terminal may ignore them. The MHP terminal may ignore possible other taps in the list.	DVB
BIOP::ModuleInfo::UserInfo	The userInfo field contains a loop of descriptors. These are specified in the DVB Data Broadcasting standard and/or this specification. The MHP terminal shall support the compressed_module_descriptor (tag 0x09) used to signal that the module is transmitted in compressed form. The userInfo field may also contain a caching_priority_descriptor and one or more label_descriptors.	DVB / This spec.

**Table B.7 : BIOP::ModuleInfo syntax**

Syntax	bits	Type	Value	Comment
BIOP::ModuleInfo() { moduleTimeOut blockTimeOut minBlockTime taps_count { id use assocTag selector_length } for (j=1; j<N1; j++) { id use assocTag selector_length for (j=0; j<N2; j++) { selector_data } } userInfoLength for (k=0; k<N3; j++) { userInfo_data } }	32 32 32 8 16 16 16 8 16 16 16 8 8 8 8	uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf	+ + + N1 0x0000 0x0017 + 0x00 + + + N2 + N3 +	≥ 1 user private BIOP_OBJECT_USE Possible additional taps that may be ignored by MHP terminals.

#### B.2.2.4.1 Label descriptor

The label\_descriptor may be placed in the userInfo field of the moduleInfo structure. It attaches a label to the corresponding module. Multiple labels can be attached to a module by including multiple label descriptors in the same userInfo field. Labels can be used for pre-fetching modules (see 10.8.3.2, "Pre-fetch descriptor" on page 228).

Within one object carousel, the same label may not be used in multiple DII messages. This implies that all modules that share a label are signalled in the same DII message.

**Table B.8 : Label descriptor syntax**

Syntax	bits	Type	Value	Comment
<pre>label_descriptor() {   descriptor_tag   descriptor_length   for (n=0; n&lt;N1; n++) {     text_char   } }</pre>	8	uimsbf	0x70	The label
	8	uimsbf	N1	
	8	uimsbf		

**descriptor\_tag:** This 8 bit integer value with 0x70 identifies this descriptor.

**text\_char:** The label that is attached to the module

#### B.2.2.4.2 Caching priority descriptor

To indicate priorities for the objects, a caching\_priority\_descriptor may be included in the userInfo field of the moduleInfo in the DownloadInfoIndication message.

This descriptor provides a priority value for the caching. The same priority applies for each object in the module. The priority indicated in the descriptor is only a hint to the MHP terminal and implementations may use that in combination with other caching strategies.

The descriptor includes also the transparency level (see section B.5.2, "Transparency levels of caching" on page 417) that shall be used by the terminal implementation if it caches objects in this module.

**Table B.9 : Caching priority descriptor syntax**

Syntax	bits	Type	Value	Comment
<pre>caching_priority_descriptor() {   descriptor_tag   descriptor_length   priority_value   transparency_level }</pre>	8	uimsbf	0x71	
	8	uimsbf		
	8	uimsbf		
	8	uimsbf		

**descriptor\_tag:** This 8 bit integer value with 0x71 identifies this descriptor.

**priority\_value:** indicates the caching priority for the objects within this module. A higher value indicates more importance for caching.

**transparency\_level:** Transparency level that shall be used by the MHP terminal if it caches objects contained in this module. The possible values are listed in table B.10. The semantics of the policies are defined in section B.5.2, "Transparency levels of caching" on page 417.

**Table B.10 : Transparency level values**

Value	Description
0	reserved
1	Transparent caching
2	Semi-transparent caching
3	Static caching.
4...255	reserved for future use

When this descriptor is not included in the userInfo field of the moduleInfo for a module, the default values that shall be assumed are:

- priority\_value: 128
- transparency\_level: 1 (transparent caching)

### B.2.2.5 ServiceGatewayInfo

The ServiceGatewayInfo structure is carried in the DownloadServerInitiate message and provides the object reference to the ServiceGateway object.

**Table B.11 : Restrictions on the ServiceGatewayInfo**

Field	Restrictions	Source
BIOP:: ServiceGatewayInfo:: downloadTaps	The MHP terminal may ignore the downloadTap list.	This spec.
BIOP:: ServiceGatewayInfo:: serviceContextList	The MHP terminal may ignore the service context list.	This spec.
BIOP:: ServiceGatewayInfo:: UserInfo	The MHP terminal may ignore the user info.	This spec.

**Table B.12 : ServiceGatewayInfo() syntax**

Syntax	bits	Type	Value	Comment
ServiceGatewayInfo() { IOP::IOR() downloadTaps_count	8	uimsbf	+ N1	See <a href="#">Table B.21 on page 397</a> software download Taps
for (i=0; i<N1; i++) { DSM::Tap() } serviceContextList_count	8	uimsbf	N2	serviceContextList
for (i=0; i<N2; i++) { context_id	32	uimsbf	N3	
context_data_length	16	uimsbf		
for (j=0; j<N3; j++) { context_data_byte	8	uimsbf	+	
} } userInfoLength	16	uimsbf	N5	user info
for (i=0; i<N5; i++) { userInfo_data	8	uimsbf	+	
} }				

### B.2.2.6 Download Cancel

There is no semantic for this message in this profile. Receivers may ignore them.

## B.2.3 The Object Carousel

### B.2.3.1 BIOP Generic Object Message

The BIOP Generic Object Message is a common structure used by all the BIOP (Broadcast Inter-ORB Protocol) messages.

**Table B.13 : Restrictions on the BIOP Generic Object Message**

Field	Restrictions	Source
MessageHeader::byte_order	0 (indicating big-endian byte order)	DVB
MessageSubHeader::objectKey	Maximum length of the key shall be four bytes.	DVB
MessageSubHeader::objectKind	The short three-letter aliases shall be used, plus the null-terminator.	DVB
Access attributes	Access attributes are not transmitted in object carousels	DSM-CC

### B.2.3.2 CORBA strings

In a number of places Object Carousel messages include text strings. These are formatted in accordance with 12.3.2 of [CORBA/IIOP \[2\]](#) and using CDR-Lite encoding as specified by DSM-CC. I.e. the text is preceded by an integer specifying the length of the string and followed by a null terminator. The size of this integer depends on the string concerned and can be seen clearly in the syntax tables that follow. However, for clarity CORBA format strings and the size of their length fields are summarised in table B.14:

**Table B.14 : Location of CORBA format strings**

string	length field size (bits)	location
objectKind_data	8	<a href="#">Table B.16, "BIOP::FileMessage syntax," on page 394</a>
objectKind_data id_data kind_data	32 8 8	<a href="#">Table B.19, "BIOP::DirectoryMessage syntax," on page 395</a>
objectKind_data	8	<a href="#">Table B.28, "BIOP::StreamMessage syntax," on page 401</a>
objectKind_data eventName_data	32 8	<a href="#">Table B.30, "BIOP::StreamEventMessage syntax," on page 403</a>
type_id_byte	32	<a href="#">Table B.21, "IOP::IOR syntax," on page 397</a>
id_data kind_data	32 32	<a href="#">Table B.25, "Syntax of Lite Options Profile Body with ServiceLocation component.," on page 400</a>

### B.2.3.3 BIOP FileMessage

The BIOP FileMessage is used for carrying file objects.

**Table B.15 : Restrictions on the BIOP File Message**

Field	Restrictions	Source
MessageSubHeader::ObjectInfo	The first 8 bytes of the ObjectInfo contain the ContentSize attribute. This is optionally followed by a loop of descriptors. The descriptors defined for possible use in this location are: <a href="#">Content type descriptor</a>	This spec.
MessageSubHeader::ServiceContextList	The MHP terminal may skip the possible serviceContextList structures.	This spec.

Table B.16 : BIOP::FileMessage syntax

Syntax	bits	Type	Value	Comment
BIOP::FileMessage() {				
magic	4x8	uimsbf	0x42494F50	"BIOP"
biop_version.major	8	uimsbf	0x01	BIOP major version 1
biop_version.minor	8	uimsbf	0x00	BIOP minor version 0
byte_order	8	uimsbf	0x00	Big endian byte ordering
message_type	8	uimsbf	0x00	
message_size	32	uimsbf	*	
objectKey_length	8	uimsbf	N1	<= 4
for (i=0; i<N1; i++) {				
objectKey_data	8	uimsbf	+	
}				
objectKind_length	32	uimsbf	0x00000004	
objectKind_data	4x8	uimsbf	0x66696C00	"fil" type_id alias
objectInfo_length	16	uimsbf	N2	
DSM::File::ContentSize	64	uimsbf	+	objectInfo
for (i=0; i<N2-8; i++) {				
descriptor()	8	uimsbf	+	
}				
serviceContextList_count	8	uimsbf	N3	serviceContextList
for (i=0; i<N3; i++) {				
context_id	32	uimsbf		
context_data_length	16	uimsbf	N4	
for (j=0; j<N4; j++) {				
context_data_byte	8	uimsbf	+	
}				
}				
messageBody_length	32	uimsbf	*	
content_length	32	uimsbf	N5	
for (i=0; i<N5; i++) {				
content_byte	8	uimsbf	+	actual file content
}				
}				

#### B.2.3.4 Content type descriptor

Zero or one content type descriptors can be carried in the file `MessageSubHeader::ObjectInfo` or the `BIOP::Binding::ObjectInfo`. Where more than one content type descriptor is used they shall express the same content format. Also, the content type (if any) signalled in the directory binding shall be identical to that signalled in the bound file's header. This optional descriptor identifies the media type of the file.

This content type signalling only applies to objects of type file and is not appropriate for other object types.

If this descriptor is absent or not sufficient to categorise the content type then the extension portion of the file name should be used to provide the media type mapping via table 6, "File type identification" on page 70.

The format of the content type descriptor is shown in table B.17.

**Table B.17 : Content type descriptor syntax**

Syntax	bits	Type	Value	Comment
content_type_descriptor() { descriptor_tag descriptor_length for (i=0; i<descriptor_length; i++) { content_type_data_byte } }	8 8 8	uimsbf uimsbf uimsbf	0x72	A MIME type

**descriptor\_tag:** This 8-bit integer with value 0x72 identifies this descriptor.

**descriptor\_length:** This 8-bit integer identifies the number of bytes following it.

**content\_type\_data\_byte:** These bytes form a string that indicates the MIME content type of the object. The string is specified as follows:

```
content_type_data = type "/" subtype *(";" parameter)
```

Where *type*, *subtype* and *parameter* are as defined in section 5 of RFC 2045 [64] and hence *content\_type\_data* carries the payload of the Content-Type header defined in [64].

### B.2.3.5 BIOP DirectoryMessage

The BIOP DirectoryMessage is used for carrying the directory objects.

**Table B.18 : Restrictions on the BIOP Directory Message**

Field	Restrictions	Source
MessageSubHeader:: ObjectInfo	The MHP terminal may skip the N2 possible bytes in the objectInfo field.	This spec.
MessageSubHeader:: ServiceContextList	The MHP terminal may skip the N3 possible serviceContextList structures.	This spec.
BIOP::Name	The name shall contain exactly one NameComponent.	This spec.
BIOP::Binding:: BindingType	Either "ncontext" (in the case of a Directory object) or "nobject" (in the case of a File or a Stream object). Binding type "composite" shall not be used.	DVB
BIOP::Binding:: ObjectInfo	After any DSM-CC defined elements the ObjectInfo for bound objects can carry an optional descriptor loop. Where the bound object is a file, DSM-CC requires the first element to be DSM::File::ContentSize after which the descriptors can be placed. The descriptors defined for possible use in this location are: <a href="#">Content type descriptor</a>	This spec.

**Table B.19 : BIOP::DirectoryMessage syntax (Sheet 1 of 3)**

Syntax	bits	Type	Value	Comment
BIOP::DirectoryMessage() { magic biop_version.major biop_version.minor byte_order message_type message_size objectKey_length for (i=0; i<N1; i++) {	4x8 8 8 8 8 32 8	uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf	0x42494F50 0x01 0x00 0x00 0x00 * N1	"BIOP" BIOP major version 1 BIOP minor version 0 big endian byte ordering      ≤ 4

Table B.19 : BIOP::DirectoryMessage syntax (Sheet 2 of 3)

Syntax	bits	Type	Value	Comment
objectKey_data	8	uimsbf	+	
objectKind_length	32	uimsbf	0x00000004	
objectKind_data	4x8	uimsbf	0x64697200	"dir" type_id alias
objectInfo_length	16	uimsbf	N2 = 0 (note 1)	objectInfo
for (i=0; i<N2; i++) { objectInfo_data }	8	uimsbf	+	
serviceContextList_count	8	uimsbf	N3	serviceContextList
for (i=0; i<N3; i++) { context_id context_data_length for (j=0; j<N4; j++) { context_data_byte } }	32 16 8	uimsbf uimsbf uimsbf	N4 +	
messageBody_length	32	uimsbf	*	
bindings_count	16	uimsbf	N5	Binding
for (i=0; i<N5; i++) { BIOP::Name() { nameComponents_count	8	uimsbf	N6 = 1	See Table B.15.
for (i=0; i<N6; i++) { id_length	8	uimsbf	N7	NameComponent id
for (j=0; j<N7; j++) { id_data	8	uimsbf	+	
} kind_length	8	uimsbf	N8	NameComponent kind
for (j=0; j<N8; j++) { kind_data	8	uimsbf	+	as type_id (see Table 4-4 in TR 101 202)
} } } BindingType	8	uimsbf	+	0x01 for nobject 0x02 for ncontext
IOP::IOR()			+	objectRef see Table B.21 on page 397
objectInfo_length	16	uimsbf	N9	
if(kind_data == "fil"){ DSM::File::ContentSize	64	uimsbf	+	0 means that file size is not signalled
for (j=0; j<N9-8; j++) { descriptor()			+	
} } else { for (j=0; j<N9; j++) { descriptor()			+	
} } }				



**Table B.19 : BIOP::DirectoryMessage syntax (Sheet 3 of 3)**

Syntax	bits	Type	Value	Comment
}				
NOTE 1: See item 2 under 11.3.2.2 "Directory Message Format" in DSM-CC "the objectInfo field shall be empty".				

### B.2.3.6 BIOP ServiceGateway message

The syntax of the BIOP ServiceGateway message is identical to that of the [BIOP DirectoryMessage](#) (described above) with the following exceptions:

- the object kind is "srg" rather than "dir".

### B.2.3.7 BIOP Interoperable Object References

The Interoperable Object References (IOR) are references to objects and contain the necessary information to locate the object. The IOR structure may contain different options to be able to point to objects that can be reached via different types of connections. For this specification, the use of IORs is limited to references to objects carried in broadcast object carousels. For object carousels, there are two types of object references: one to be used to reference objects carried in the same object carousel and one to be used to reference objects in other object carousels.

**Table B.20 : Restrictions on the BIOP IOR**

Field	Restrictions	Source
IOP::IOR::type_id	Contains the objectKind of the referenced object. A short three-letter aliases shall be used, plus a null-terminator.	This spec.
IOP::IOR::taggedProfileList	There shall be at least 1 taggedProfile included in an IOR. For objects carried in a broadcast object carousel, the first taggedProfile shall be either a TAG_BIOP profile or a TAG_LITE_OPTIONS. If the first tagged profile is some other profile, the object is not carried in a broadcast object carousel and the MHP terminal may ignore the object subject to its own capabilities.	This spec.

**Table B.21 : IOP::IOR syntax**

Syntax	bits	Type	Value	Comment
IOP::IOR { type_id_length for (i=0; i<N1; i++) { type_id_byte } taggedProfiles_count IOP::taggedProfile()  for (n=0; n<N2-1;n++) { IOP::taggedProfile() } }	32	uimsbf	N1	
	8	uimsbf	+	Short alias type_id (e.g. "dir")
	32	uimsbf	N2	Profile bodies For objects in broadcast carousels: either <a href="#">BIOPProfileBody</a> or <a href="#">LiteOptionsProfileBody</a> .
				MHP terminal may ignore other profiles (2...N1) if present

## B.2.3.7.1 BIOPProfileBody

The BiopProfileBody is used for references to objects within the same object carousel.

Table B.22 : Restrictions on the BIOP Profile Body

Field	Restrictions	Source
BiopProfileBody::byte_order	0 (indicating big-endian byte order)	DVB
BiopProfileBody::LiteComponent	The list shall contain exactly 1 BiopObjectLocation and exactly 1 DSM::ConnBinder as the first two components in that order. The MHP terminal may ignore possible other components in the list.	This spec.
DSM::ConnBinder	For objects carried in the broadcast object carousel, the first Tap shall be of type BIOP_DELIVERY_PARA_USE. If there is another type of tap in the first position, the MHP terminal may ignore this object reference, as it is a reference for object accessed using another type of protocol (e.g. for return channel use). The MHP terminal may ignore possible other taps in the list.	This spec.
DSM::Tap	In the BIOP_DELIVER_PARA_USE tap, the id field is not used and may be ignored by the MHP terminal.	This spec.

Table B.23 : BIOP Profile Body syntax (Sheet 1 of 2)

Syntax	bits	Type	Value	Comment
BIOPProfileBody {				
profileId_tag	32	uimsbf	0x49534F06	TAG_BIOP (BIOP Profile Body)
profile_data_length	32	uimsbf	*	
profile_data_byte_order	8	uimsbf	0x00	big endian byte order
lite_component_count	8	uimsbf	N1	
BIOP::ObjectLocation {				
componentId_tag	32	uimsbf	0x49534F50	TAG_ObjectLocation
component_data_length	8	uimsbf	*	
carouselId	32	uimsbf	+	
moduleId	16	uimsbf	+	
version.major	8	uimsbf	0x01	BIOP protocol major version 1
version.minor	8	uimsbf	0x00	BIOP protocol minor version 0
objectKey_length	8	uimsbf	N2	<= 4
for (k=0; k<N2; k++) {				
objectKey_data	8	uimsbf	+	
}				
}				
DSM::ConnBinder {				
componentId_tag	32	uimsbf	0x49534F40	TAG_ConnBinder
component_data_length	8	uimsbf	N4	
taps_count	8	uimsbf	N3	
DSM::Tap {				
id	16	uimsbf	0x0000	user private

Table B.23 : BIOP Profile Body syntax (Sheet 2 of 2)

Syntax	bits	Type	Value	Comment
<pre> use </pre>	16	uimsbf	0x0016	<p>If BIOP_DELIVERY_PARA_USE is provided it shall be the first tap.</p> <p>If there is another type of tap in the first position, the MHP terminal may ignore this object reference, as it is a reference for an object accessed using another type of protocol (e.g. for return channel use).</p>
<pre>     assocTag </pre>	16	uimsbf	+	
<pre>     selector_length </pre>	8	uimsbf	0x0A	
<pre>     selector_type </pre>	16	uimsbf	0x0001	
<pre>     transactionId </pre>	32	uimsbf	*	
<pre>     timeout   } </pre>	32	uimsbf	*	
<pre> for (n=0; n&lt;N4-18; n++) { </pre>				<p>The MHP terminal may skip over the possible additional taps</p> <p>N6=N1-2</p>
<pre>     additional_tap_byte   } </pre>	8	uimsbf		
<pre> } for (n=0;n&lt;N6;n++) {   BIOP::LiteComponent{ </pre>				
<pre>     componentId_tag </pre>	32	uimsbf	+	
<pre>     component_data_length </pre>	8	uimsbf	N7	
<pre>     for (i=0; i&lt;N7; i++) {       component_data_byte     }   } } } </pre>	8	uimsbf		

### B.2.3.7.2 LiteOptionsProfileBody

The LiteOptionsProfileBody is used for making links to objects carried in other object carousels. The LiteOptionsProfileBody can be used to make references to objects carried in other carousels within the same transport or in other transport streams. In the API, when the LiteOptionsProfileBody is encountered, the application will get a ServiceXFRErrorEvent or a ServiceXFRException. The other carousel is never mounted automatically by the implementation, but the application may do so using the API.

Table B.24 : Restrictions on the Lite Options Profile Body

Field	Restrictions	Source
LiteOptionsProfileBody::profile_data_byte_order	0 (indicating big-endian byte order)	DVB
LiteOptionsProfileBody::LiteOptionComponents	The list shall contain a ServiceLocation component as the first component. The MHP terminal may ignore possible other components in the list.	This spec.
DSM::ServiceLocation	For objects carried in the broadcast object carousel, the service domain NSAP address shall follow the Carousel NSAP address format.	This spec.
DSM::ServiceLocation::InitialContext	The MHP terminal may ignore the initial context	This spec.

**Table B.25 : Syntax of Lite Options Profile Body with ServiceLocation component.**

Syntax	bits	Type	Value	Comment
LiteOptionsProfileBody {				
profileId_tag	32	uimsbf	0x49534F05	TAG_LITE_OPTIONS (Lite Options Profile Body)
profile_data_length	32	uimsbf	*	
profile_data_byte_order	8	uimsbf	0x00	big endian byte order
lite_component_count	8	uimsbf	N1	
DSM::ServiceLocation {				
componentId_tag	32	uimsbf	0x49534F46	TAG_ServiceLocation
component_data_length	8	uimsbf	*	
serviceDomain_length	8	uimsbf	0x14	Length of carousel NSAP address
serviceDomain_data()	160	uimsbf	+	Table B.26 "DVB Carousel NSAP Address" pathName
CosNaming::Name() {				
nameComponents_count	32	uimsbf	N2	
for (i=0; i<N2; i++) {				
id_length	32	uimsbf	N3	NameComponent id
for (j=0; j<N3 j++) {				
id_data	8	uimsbf	+	
}				
kind_length	32	uimsbf	N4	NameComponent kind
for (j=0; j<N4 j++) {				
kind_data	8	uimsbf	+	as type_id (see Table 4-4 in TR 101 202)
}				
}				
initialContext_length	32	uimsbf	N5	
for (n=0; n<N5 n++) {				
InitialContext_data_byte	8	uimsbf		
}				
for (n=0;n<N6;n++) {				N6=N1-1
BIOP::LiteComponent{				
componentId_tag	32	uimsbf	+	
component_data_length	8	uimsbf	N7	
for (i=0; i<N7; i++) {				
component_data_byte	8	uimsbf		
}				
}				
}				
}				

**Table B.26 : DVB Carousel NSAP Address (Sheet 1 of 2)**

Syntax	bits	Type	Value	Comment
DVBcarouselNSAPaddress {				
AFI	8	uimsbf	0x00	NSAP for private use
Type	8	uimsbf	0x00	Object carousel NSAP Address.

Table B.26 : DVB Carousel NSAP Address (Sheet 2 of 2)

Syntax	bits	Type	Value	Comment
carouselId	32	uimsbf	+	To resolve this reference a carousel_id_descriptor with the same carousel_id as indicated in this field must be present in the PMT signalling for the service identified below.
specifierType	8	uimsbf	0x01	IEEE OUI
specifierData { IEEE OUI }	24	uimsbf	0x00015A	Constant for DVB OUI
dvb_service_location () { transport_stream_id	16	uimsbf	+	This may be set to 0x0000 which indicates that the MHP terminal shall not use the transport_stream_id when locating the service. For any other value then this field shall be used.
original_network_id	16	uimsbf	+	(= MPEG-2 program_number)
service_id	16	uimsbf	+	
reserved	32	bslbf	0xFFFFFFFF	
}				

## B.2.3.8 BIOP StreamMessage

Table B.27 : Restrictions on the BIOP Stream Message

Field	Restrictions	Source
MessageSubHeader:: ObjectInfo	The ObjectInfo field contains the DSM::Stream::Info_T structure and optionally other data after the Stream Info structure. MHP terminals may ignore the DSM::Stream::Info_T structure and the possible other object info data following it. Broadcasts may set the duration field to zero to indicate undefined duration.	This spec.
MessageSubHeader:: ServiceContextList	The MHP terminal may skip the possible serviceContextList structures.	This spec.
MessageSubHeader:: MessageBody	The MessageBody carries a sequence of taps. There shall be at most one tap of use BIOP_PROGRAM_USE. This tap identifies the service that provides the media stream associated with the Stream object (via a deferred_association_tags_descriptor in the PMT). The tap may only reference programs that are broadcast on the same multiplex (i.e. MHP terminals shall not need to tune to a different multiplex in order to receive the referenced media stream). There shall also be at most one tap with use STR_NPT_USE, which MHP terminals shall interpret as described in <a href="#">ISO/IEC 13818-6 [26]</a> . MHP terminals may ignore possible other Taps (such as BIOP_ES_USE).	This spec.

Table B.28 : BIOP::StreamMessage syntax (Sheet 1 of 2)

Syntax	bits	Type	Value	Comment
BIOP::StreamMessage() {				
magic	4x8	uimsbf	0x42494F50	"BIOP"
biop_version.major	8	uimsbf	0x01	BIOP major version 1
biop_version.minor	8	uimsbf	0x00	BIOP minor version 0
byte_order	8	uimsbf	0x00	big endian byte ordering
message_type	8	uimsbf	0x00	
message_size	32	uimsbf	*	
objectKey_length	8	uimsbf	N1	<= 4
for (i=0; i<N1; i++) {				
objectKey_data	8	uimsbf	+	



## B.2.3.9 BIOP StreamEventMessage

Table B.29 : Restrictions on the BIOP StreamEvent Message

Field	Restrictions	Source
MessageSubHeader::ObjectInfo	The ObjectInfo field contains the DSM::Stream::Info_T and DSM::Stream::EventList_T structures followed optionally by other object info data (which may be ignored by MHP terminals). See <a href="#">Table B.27 on page 401</a> regarding the DSM::Stream::Info_T. MHP terminals may ignore the possible other data following the DSM::Stream::EventList_T. The EventList_T defines a sequence of event names that correlates to the sequence of event ids in the MessageBody. eventNames_count shall equal eventIds_count.	This spec.
MessageSubHeader::ServiceContextList	The MHP terminal may skip the possible serviceContextList structures.	This spec.
MessageSubHeader::MessageBody	The MessageBody carries a sequence of taps followed by a sequence of event ids. The sequence of taps follows the following rules: <ul style="list-style-type: none"> <li>• There shall be at most one tap of use BIOP_PROGRAM_USE. This tap identifies the service that provides the media stream associated with the Stream object (via a deferred_association_tags_descriptor in the PMT). The tap may only reference programs that are broadcast on the same multiplex (i.e. MHP terminals shall not need to tune to a different multiplex in order to receive the referenced media stream).</li> <li>• There shall be at most one tap with use STR_NPT_USE, which MHP terminals shall interpret as described in <a href="#">ISO/IEC 13818-6 [26]</a>.</li> <li>• There shall be at most one tap with use STR_EVENT_USE or STR_STATUS_AND_EVENT_USE. This tap indicates the PID where all StreamEvent descriptors related to the StreamEvent object are broadcast. MHP terminals may ignore possible other Taps (such as BIOP_ES_USE).</li> </ul>	This spec.

Table B.30 : BIOP::StreamEventMessage syntax (Sheet 1 of 2)

Syntax	bits	Type	Value	Comment
BIOP::StreamEventMessage() {				
magic	4x8	uimsbf	0x42494F50	"BIOP"
biop_version.major	8	uimsbf	0x01	BIOP major version 1
biop_version.minor	8	uimsbf	0x00	BIOP minor version 0
byte_order	8	uimsbf	0x00	big endian byte ordering
message_type	8	uimsbf	0x00	
message_size	32	uimsbf	*	
objectKey_length	8	uimsbf	N1	
for (i=0; i<N1; i++) {				
objectKey_data	8	uimsbf	+	
}				
objectKind_length	32	uimsbf	0x00000004	
objectKind_data	4x8	uimsbf	0x73746500	"ste" type_id alias
objectInfo_length	16	uimsbf	N2	
DSM::Stream::Info_T {				
aDescription_length	8	uimsbf	N3	aDescription
for (i=0; i<N3; i++) {				
aDescription_bytes	8	uimsbf	+	see <a href="#">BIOP StreamMessage</a>
}				
duration.aSeconds	32	simsbf	+	see <a href="#">BIOP StreamMessage</a>
duration.aMicroSeconds	32	uimsbf	+	see <a href="#">BIOP StreamMessage</a>
audio	8	uimsbf	+	see <a href="#">BIOP StreamMessage</a>
video	8	uimsbf	+	see <a href="#">BIOP StreamMessage</a>

Table B.30 : BIOP::StreamEventMessage syntax (Sheet 2 of 2)

Syntax	bits	Type	Value	Comment
data	8	uimsbf	+	see BIOP StreamMessage
} DSM::Event::EventList_T { eventNames_count	16	uimsbf	N4	(including zero terminator)
for (i=0; i<N4; i++) { eventName_length	8	uimsbf	N5	
for (j=0; j<N5; j++) { eventName_data	8	uimsbf	+	
} } } for (i=0; i<N2-(N3+10); i++) { objectInfo_byte	8	uimsbf	+	
} serviceContextList_count	8	uimsbf	N6	
for (i=0; i<N6; i++) { context_id	32	uimsbf	N7	
context_data_length	16	uimsbf		
for (j=0; j<N7; j++) { context_data_byte	8	uimsbf		
} } messageBody_length	32	uimsbf	*	see B.2.4.4 "Timebases" see Table 4-12 in DVB Guidelines for Data Broadcasting  no selector  (= eventNames_count)
taps_count	8	uimsbf	N8	
for (i=0; i<N8; i++) { id	16	uimsbf	(note 1)	
use	16	uimsbf	+	
assocTag	16	uimsbf	+	
selector_length	8	uimsbf	0x00	
} eventIds_count	8	uimsbf	N4	
for (i=0; i<N4; i++) { eventId	16	uimsbf	+	
} } }				
NOTE 1: If the tap use is STR_NPT_USE then the value of this 16 bit integer corresponds to the value of the contentId field of the NPTReferenceDescriptor that defines the time base for this stream. For other values of tap use the value of this field is undefined.				

## B.2.4 Stream Events

There are two versions of stream messages. The BIOP StreamMessage is used for carrying the stream objects that don't use DSM-CC Stream events. The BIOP StreamEventMessage is used for carrying stream objects that include a stream carrying the DSM-CC Stream events. This section addresses the later.

### B.2.4.1 Stream & StreamEvent messages

#### B.2.4.1.1 Association with time bases

The id field of the STR\_NPT\_USE tap of a StreamMessage or StreamEventMessage identifies the timebase associated with that Stream/StreamEvent object. Multiple StreamMessage or StreamEventMessage may be used at the same time to allow subscriptions to multiple timebases of the same service. See B.2.4.4, "Timebases" on page 407.



### B.2.4.1.2 Event names and event ids

In StreamEventMessages the EventList\_T defines a sequence of event names that correlates 1:1 to the sequence of event ids in the MessageBody. Within each BIOP::StreamEventMessage the event names uniquely associate to event id values.

- The eventNames\_count shall equal eventIds\_count.
- The names in the EventList\_T are zero-terminated strings.
- The eventID values in the StreamEventMessage correspond to the eventID values carried in StreamEventDescriptors.

### B.2.4.1.3 Stream event life time

In StreamEventMessages the set of events described in the BIOP::StreamEvent message is possibly a subset of the events that may be used by the application during the course of a programme. Therefore, applications may need to accommodate the dynamic change of such messages. Cache transparency (see section B.5.2.1, "Transparent caching" on page 417) and version listener mechanisms (see DSMCCObject methods in annex P, "(normative): Broadcast Transport Protocol Access" on page 624) provide applications with the means to do this.

Similarly the set of stream event descriptors being transmitted at any time may not correspond to the set of events described in the BIOP::StreamEventMessage.

The event id for an event name shall not change while the name exists. If a name is removed it shall not be reintroduced with within 60 seconds.

## B.2.4.2 Stream Descriptors

### B.2.4.2.1 NPT Reference descriptor

#### B.2.4.2.1.1 Usage scenarios

The following 3 usage scenarios are envisaged for NPT:

- No NPT for associated with a service  
Enables use of "do it now" events but not scheduled events
- A single continuous timebase (i.e. a single progressing value of NPT)  
In this case all types of stream event can be used. However, the broadcast is logically a single continuing interactive production and the broadcaster is responsible for preprocessing the applications etc. before broadcast to be suitable
- The signal received by the MHP terminal includes a unique timebase for each programme needing one. This timebase to be suspended during any insertion into that programme and this timebase to be discontinued at the end of the programme.

i.e. the timebase for each programme is preserved through the distribution network.

#### B.2.4.2.1.2 Syntax

MHP terminals shall interpret this descriptor as it is described in ISO/IEC 13818-6 [26] with the following clarifications and additions.

With regard to contentId:

- This 7 bit field identifies the "timebase" see B.2.4.4, "Timebases" on page 407.

With regard to scaleNumerator and scaleDenominator:

- 1/1 - means normal play.  
i.e. NPT and STC advance at the same rate.
- 0/m (m > 0) - means the NPT value does not advance.

As a consequence no scheduled stream events will be raised. However, the "do it now" events continue to be effective.

- 0/0 - means that the scaleNumerator and scaleDenominator fields are not defined in the NPT Reference descriptor and should be derived as described in [ISO/IEC 13818-6 \[26\]](#) "8.1.2 Reconstruction of NPT"

MHP terminals are not required to support this mode of operation.

- m/0 ( $m > 0$ ) - this is not allowed by [ISO/IEC 13818-6 \[26\]](#).

With regard to broadcast repetition rate:

- NPT Reference descriptors shall be transmitted at least once per second.

#### B.2.4.2.2 Stream event descriptor

##### B.2.4.2.2.1 Association of event ids to event time

The eventNPT field conveys the NPT value at which the event will occur (or has occurred).

Each StreamEventDescriptor provides a single association between an eventID and a value of eventNPT. If the MHP terminal detects a change in the value of eventNPT associated with a value of eventID this redefines the time at which the event should fire.

MHP terminals shall ignore scheduled events where the eventNPT has passed.

See also "number range for NPT" on page 406 and "Signalling of "do it now events"" on page 406.

##### B.2.4.2.2.2 Re-use of event ids

Event ID values may be re-used any number of times. For example, after an event has fired then stream event descriptors with the same eventID but different eventNPT may be broadcast.

##### B.2.4.2.2.3 Signalling of "do it now events"

[ISO/IEC 13818-6 \[26\]](#) is silent on the broadcast signalling of "do it now" events.

These events shall be identified by the value of eventID and hence table id extension (see "Encoding of table id extension" on page 407).

Where the value of eventID identifies a "do it now" event then the value of eventNPT shall be ignored by the MHP terminal.

##### B.2.4.2.2.4 Private data

The privateDataByte field does not need to be interpreted by the MHP terminal.

- NOTE: that an application can access the privateDataByte field via [11.4.2.5, "Extensions to the Framework" on page 251](#) and [11.5.1, "Broadcast Transport Protocol Access API" on page 255](#)

#### B.2.4.2.3 Unused descriptors

MHP terminals may ignore the following descriptors if present:

- NPT Endpoint descriptor
- Stream Mode descriptor

#### B.2.4.2.4 Clarification of number encoding

##### B.2.4.2.4.1 number range for NPT

There is some ambiguity in [ISO/IEC 13818-6 \[26\]](#) regarding the data type used to carry NPT values in the signalling (tcimsbf or uimsbf). The following requirements insulate this profile from this ambiguity:

- The range of values used shall be in the range 0 to 0x0FFFFFFF (which is unambiguous for both tcimsbf or uimsbf).

#### B.2.4.2.4.2 number range for scaleDenominator

There is some ambiguity in ISO/IEC 13818-6 [26] regarding the data type used to carry scaleDenominator values in the signalling (tcimsbf or uimsbf). The following requirements insulate this profile from this ambiguity:

- The range of values used shall be in the range 0 to 0x7FFF (which is unambiguous for both tcimsbf or uimsbf).

#### B.2.4.3 DSM-CC Sections carrying Stream Descriptors

##### B.2.4.3.1 Section version number

The section version number field increments to reflect changes in stream descriptor(s) carried by sections with the same value of table\_id (0x3D) and table\_id\_extension.

The version number shall increment for reasons including the change in value of eventNPT for a given eventId.

##### B.2.4.3.2 Single firing of "do it now" events

MHP terminals shall respond at most one time to a "do it now" events delivered under a single table version.

##### B.2.4.3.3 Section number

For this specification MHP terminals shall only consider section number zero.

##### B.2.4.3.4 DSM-CC sections for DSMCC\_descriptor\_list()

If the table\_id field equals 0x3D the current\_next\_indicator bit shall be set to "1".

##### B.2.4.3.5 Encoding of table id extension

The section's table id extension field provides information on the stream descriptor(s) carried by the section:

**Table B.31 : Encoding of table id extension for DSMCC\_descriptor\_lists**

table_id_extension bits			Payload of DSM-CC section with table ID 0x3D
[15]	[14]	[13...0]	
0	0	eventID[13...0]	Section carries a single "do it now" event
0	1	xx xxxx xxxx	Section carries a single NPT reference descriptor
1	0	xx xxxx xxxx	Section carries one or more other stream descriptors. I.e - Stream event descriptor(s) with a future eventNPTs - Stream mode descriptor (can be ignored in this profile) - NPT endpoint descriptor (can be ignored in this specification)
1	1	reserved for future use	

The value of eventID for "do it now" events shall be in the range 0x0001...0x3FFF. The value of eventID for scheduled events shall be in the range 0x8000...0xBFFF. The value 0 is not allowed (see 5.5.2.2.1 in ISO/IEC 13818-6 [26]).

#### B.2.4.4 Timebases

Multiple concurrent timebases may be defined for a single MPEG program but only a single time base is allowed to progress at any instant (the other timebases shall be paused). The relationship between each timebase and the MPEG timebase (STC) is defined by an NPTReferenceDescriptor. The contentId field of the NPTReferenceDescriptor (a 7 bit unsigned integer) identifies the timebase.

The value of the id field of the STR\_NPT\_USE tap (a 16 bit unsigned integer) of a StreamMessage or StreamEventMessage identifies the timebase associated with that Stream/StreamEvent object. Multiple StreamMessage or StreamEventMessage may be used at the same time to allow subscriptions to multiple timebases of the same service.

In this profile NPTReferenceDescriptors can indicate two states:

- non-paused

The scaleNumerator and scaleDenominator are both non-zero

- paused

The scaleNumerator is zero and the scaleDenominator is non-zero

When a timebase is signalled as paused then the NPT value for that timebase is frozen at NPT\_Reference (as specified by equation 8-4 in DSM-CC).

All of the NPTReferenceDescriptors for all of the timebases shall be carried in a single DSMCC\_descriptor\_list section and shall be transmitted at least once every second. In any such set of NPTReferenceDescriptors at most one shall be non-paused and there shall be at most one instance of each value of contentId.

Timebases can be added or subtracted from the current set. The set of current timebases can be empty.

A stream (i.e. a StreamMessage or StreamEventMessage) may have no associated timebase (i.e. it may have no tap with use STR\_NPT\_USE). This is valid in the following cases:

- only "do it now" events are used
- no stream events are used

A value of contentId shall not be reused for a new timebase within 60 seconds of the removal of the timebase.

In normal use the NPT of a non-paused timebase progresses at a constant rate. Discontinuities should either be the results of errors in the broadcast or transient conditions (for example, while an NPT reference generator catches up with an MPEG PCR discontinuity). Transient discontinuities should be tolerated by the MHP Terminal. The behaviour of the MHP terminal when subject to a permanent discontinuity is not specified.

The broadcaster shall start generating corrected NPTReferenceDescriptors within at most 1 second of a PCR discontinuity (ideally the descriptors should be generated before the PCR discontinuity). If the receiver is sampling the NPTReferenceDescriptor at the lowest allowed rate (once every 5 seconds) then the receiver may not receive a correct NPTReferenceDescriptor for 5 seconds. During this period the receiver should linearly extrapolate the NPT from previous NPT values in the expectation that a corrected NPTReferenceDescriptor will be delivered shortly.

There is a window of uncertainty around a segment of paused timebase due to the time taken for all receivers to acquire the new NPTReferenceDescriptor. During this window scheduled events cannot be used reliably.

NOTE: It is suggested that broadcasters use "do it now" events near junctions between different timebases.

## B.2.4.5 Monitoring stream events

### B.2.4.5.1 NPT reference monitoring

When supporting timebase stimulated events the MHP terminal shall allocate resources sufficient to ensure that updates to the set of timebases is detected within 5 seconds for conformant broadcasts.

### B.2.4.5.2 Timebase stimulated event monitoring

When supporting timebase stimulated events the MHP terminal shall allocate resources sufficient to ensure that updates to the set of timebase stimulated events is detected within 5 seconds for conformant broadcasts. So, if an event is introduced or the NPT time at which it is specified to fire is changed then the MHP terminal will respect this change within 5 seconds. If the fire time for an event changes less than 5 seconds before it was previously scheduled to fire then there is no guarantee that all receivers will detect the change in time.

If a timebase is deleted (reference to it is removed from the set of NPTReferenceDescriptors) then the receiver shall deactivate any event listeners dependant on that timebase and may free resources associated with those listeners.

### B.2.4.5.3 "do it now" events

"do it now" events are single shot events, accordingly MHP terminals need to make special efforts to ensure a high probability that they can be reliably received.

Broadcasters are responsible for placing all "do it now" stream descriptors that may be of interest to an application on a single PID. This may be the same PID as is used for other DSM-CC sections.

MHP terminals shall dedicate a section filter to monitoring the possible transmission of "do it now" events while there are any applications subscribed to these events.

#### B.2.4.5.4 scheduled events

The stream descriptors for scheduled events are transmitted several times in the period before the time that they should fire. This allows a high probability that they will be effective even if they are not monitored continuously by the MHP terminal.

Any scheduled stream event descriptors shall be transmitted at least once each second.

MHP terminals shall raise an event in response to a scheduled stream event provided that the stream event descriptors are broadcast for at least 5 seconds before the scheduled time.

#### B.2.4.5.5 number of NPT components

The MHP terminal is only required to monitor a single NPT component. So, if events from more than one DSM-CC StreamEventMessages are subscribed to no more than one stream component shall be specified as the source of NPTReferenceDescriptors (i.e. the taps with use STR\_NPT\_USE shall be the same).

### B.2.5 Assignment and use of transactionId values

The use of the transactionId in the object carousel is inherited from its use as defined by the DSM-CC specification, and as such it can appear somewhat complex. The transactionId has a dual role, providing both identification and versioning mechanisms for control messages, i.e. DownloadInfoIndication and DownloadServerInitiate messages. The transactionId should uniquely identify a download control message within a data carousel, however it should be "incremented" whenever any field of the message is modified.

Note: The term "incremented" is used in the DSM-CC specification. Within the scope of this specification this should be interpreted as "changed".

The object carousel is carried on top of one or more data carousels. By a data carousel used below the object carousel, we mean in this specification a set of DownloadInfoIndication message transmitted on a single PID and the DownloadDataBlock messages carrying the modules described in the DownloadInfoIndication messages. The DownloadDataBlock messages may be spread on other elementary streams than the DownloadInfoIndication messages. The DownloadServerInitiate message in the context of object carousels is considered to be part of the top level of the object carousel and not associated with any data carousel.

When a module is changed, the version number of the module needs to be changed. This implies that the DownloadInfoIndication message that references the module needs to be also updated. Since the DownloadInfoIndication is updated, the transactionId needs to be also changed. However, the transactionId of the DownloadInfoIndication message is used in other messages also, but the need to change the other messages should specifically be avoided and the implications of updating a module should be limited to the module itself and the DownloadInfoIndication that references the module. Therefore, additional rules on the usage of the transactionId have been specified as follows.

The transactionId has been split up into a number of sub-fields defined in [Table B.32](#). This reflects the dual role of the transactionId (outlined above) and constraints imposed to reduce the effects of updating a module. However, to increase interoperability the assignment of the transactionId has been designed to be independent of the expected filtering in target MHP terminals.

**Table B.32 : Sub-fields of the transactionId (Sheet 1 of 2)**

Bits	Value	Sub-field	Description
0	User-defined	Updated flag	This must be toggled every time the control message is updated
1-15	User-defined	Identification	This must and can only be all zeros for the DownloadServerInitiate message. All other control messages must have one or more non-zero bit(s).

**Table B.32 : Sub-fields of the transactionId (Sheet 2 of 2)**

Bits	Value	Sub-field	Description
16-29	User-defined	Version	This must be incremented/changed every time the control message is updated.
30-31	Bit 30 - zero Bit 31 - non-zero	Originator	This is defined in the DSM-CC specification [26] as 0x02 if the transactionId has been assigned by the network - in a broadcast scenario this is implicit.

Due to the role of the transactionId as a versioning mechanism, any change to a control message will cause the transactionId of that control message to be incremented. Any change to a Module will necessitate incrementing its moduleVersion field. This change must be reflected in the corresponding field in the description of the Module in the DownloadInfoIndication message(s) that describes it. Since a field in the DownloadInfoIndication message is changed its transactionId must be incremented to indicate a new version of the message. Also, any change in the DownloadServerInitiate message implies that its transactionId must also be incremented. However, when the transactionId is divided into subfields as specified above, updating a message will change only the Version part of the transactionId while the Identification part remains the same.

Since the transactionId is used also for identifying the messages when referencing the messages in other structures, it is very desirable that these referenced would not need to be updated every time the control message is update. Therefore the following rule shall be applied when locating the messages based on the references:

**When locating a message based on the transactionId value used for referencing the message, only the Identification part (bits 1...15) shall be matched.**

Using this rule, the implications of updating a module can be limited to the module itself and the DownloadInfoIndication message describing the module. Also, this implies that if an MHP terminal wants to find out if a particular module that it has retrieved earlier has changed, it needs to filter the DownloadInfoIndication message that described that module and check if it has been changed.

## B.2.6 Mapping of objects to data carousel modules

The DSM-CC Object Carousels allow one or more objects to be carried in one module of the data carousel. In order to optimize the performance and memory requirements three additional requirements are specified:

- When mapping objects to modules of a data carousel, only closely related objects should be put into one module. Objects that are not closely related should not be put into the same module. If in the process of retrieving an object from the carousel an MHP terminal acquires a module containing multiple objects, it should attempt to cache these since the expectation should be that the other objects are related to the object requested and probably will be needed soon.
- The size of a module that contains multiple objects should not exceed 65536 bytes when decompressed<sup>1</sup>. MHP terminals complying to this specification are only required to handle modules containing multiple objects where the module size when decompressed is 65536 bytes or less. Modules containing a single file message can exceed 65536 bytes with upper size only limited by the memory resources in the MHP terminal.
- In addition to the limitations imposed by the 65536 byte limit, directory and service gateway messages are limited to 512 object bindings per message.

## B.2.7 Compression of modules

The modules may be transmitted either in uncompressed or compressed form. If the module is transmitted in compressed form, this is signalled by including the compressed\_module\_descriptor in the userInfo field of the moduleInfo in the DownloadInfoIndication message.

1. I.e. when the file has been decompressed from the file transport but before the content decoding has started.

Presence of the `compressed_module_descriptor` indicates that the data in the module has the "zlib" structure as defined in RFC 1950 [75].

Table B.33 shows the syntax of the `compressed_module_descriptor`:

**Table B.33 : compressed\_module\_descriptor**

	No. of bytes	Mnemonic	Value
<code>compressed_module_descriptor() {</code>			
<code>descriptor_tag</code>	1	uimsbf	0x09
<code>descriptor_length</code>	1	uimsbf	
<code>compression_method</code>	1	uimsbf	
<code>original_size</code>	4	uimsbf	
<code>}</code>			

Presence of the `compressed_module_descriptor` indicates that the data in the module has the "zlib" structure as defined in RFC 1950 [75]. Table B.34 shows the syntax of the ZLIB structure.

**Table B.34 : zlib structure**

	No. of bytes	Value	
<code>zlib structure() {</code>			
<code>compression_method</code>	1		
<code>flags_check</code>	1		
<code>compressed_data</code>	n		
<code>check value</code>	4		
<code>}</code>			

The MHP terminal shall support the Deflate compression algorithm as specified in RFC 1951 [76]. This is signalled setting the least significant nibble of the `compression_method` to 0x8 (i.e. `compression_method` is xxxx1000). The MHP terminal is not required to support other compression algorithms.

## B.2.8 Mounting an Object Carousel

The `ServiceGateway` object is the root directory of the file system delivered by an Object Carousel and must be acquired before any other object can be downloaded. This may be achieved by two compatible mechanisms. The signalling of which mechanisms are being supported by a broadcast is provided by the `carousel_id_descriptor`. This descriptor may be included in the second descriptor loop of a PMT corresponding to a PID on which the DSI message for an Object Carousel is broadcast, i.e. the boot-PID.

In this specification the use of the `carousel_id_descriptor` for signalling is mandatory. The consequence is that if a PMT second descriptor loop contains a `data_broadcast_id_descriptor` that provides signalling for this specification, it shall also contain a `carousel_id_descriptor`.

Note: A single PID shall only contain messages from a single Object Carousel and so only one `carousel_id_descriptor` shall be present in any second descriptor loop. However, a single service may contain more than one Object Carousel. Consequently, the `carousel_id_descriptor` may appear more than once in any single PMT.

The acquisition of the `ServiceGateway` object may be via the standard DSI-DII mechanism. This shall be supported by all broadcasts regardless of signalling in the `carousel_id_descriptor` and shall be sufficient for all MHP terminals.

See also 10.2, "Program Specific Information" on page 209.

A broadcast may also contain additional information in the `carousel_id_descriptor` to support the "enhanced" boot mechanism. This is signalled by setting the `formatId` field for this descriptor to 0x01. This additional information is an aggregation of all the fields necessary to locate the `ServiceGateway`, also found in the DSI and DII messages. However, in such a case the module containing the `ServiceGateway` object shall be broadcast on the PID identified by the `data_broadcast_id_descriptor`. It is optional for both broadcasts and MHP terminals to support this mechanism.

### B.2.8.1 carousel\_id\_descriptor

This descriptor is MPEG defined and in this specification may be included in the second descriptor loop of a PMT.

**Table B.35 : Carousel identifier descriptor syntax**

Syntax	bits	Type	Value
carousel_identifier_descriptor {			
descriptor_tag	8	uimsbf	0x13
descriptor_length	8	uimsbf	N1
carousel_id	32	uimsbf	
FormatID	8	uimsbf	
if( FormatID == 0x00 ) {			
for( i=0; i<N1-5; i++ ){			
private_data_byte	8		
}			
}			
if( FormatID == 0x01 ) {			
ModuleVersion	8	uimsbf	
ModuleId	16	uimsbf	
BlockSize	16	uimsbf	
ModuleSize	32	uimsbf	
CompressionMethod	8	uimsbf	
OriginalSize	32	uimsbf	
TimeOut	8	uimsbf	
ObjectKeyLength	8	uimsbf	N2 ≤ 4
for( i=0; i<N2; i++ ){			
ObjectKeyData	8	bslbf	
}			
for( i=0; i<N1-N2-21; i++ ){			
private_data_byte	8		
}			
}			
}			

**carousel\_id:** The 32 bit field it identifies the object carousel with the corresponding carouselId.

**FormatID:** This 8 bit integer identifies whether the carousel supports the "enhanced boot" mechanism or not. The value 0x00 indicates "standard boot", 0x01 indicates that "enhanced boot" is possible.

**ModuleVersion:** This 8 bit integer is the version number of the module containing the service gateway. This is equivalent to moduleVersion in the DII.

**ModuleId:** This 16 bit integer is the identifier of the module in the carousel. This is equivalent to moduleId in the DII.

**BlockSize:** This 16 bit integer is the size in bytes of every block in the module (except for the last block which may be the same or smaller). This is equivalent to blockSize in the DII.

**ModuleSize:** This 32 bit integer is the size of the module in bytes. This is equivalent to moduleSize in the DII.

**CompressionMethod:** This 8 bit field identifies the compression algorithm defined in [RFC 1950 \[75\]](#) used to compress the module. It is equivalent to compression\_method carried in the compressed\_module\_descriptor in the DII.

**OriginalSize:** This 32 bit integer is the size of the data (in bytes) carried by the module before it was compressed. It is equivalent to original\_size carried in the compressed\_module\_descriptor in the DII.

If the module has not been compressed the values of OriginalSize and ModuleSize shall be equal and the value of CompressionMethod is not defined.



**TimeOut:** This 8 bit integer specifies the timeout in seconds for acquisition of all blocks of the module.

**ObjectKeyLength:** This 8 bit integer specifies the number of bytes of ObjectKeyData.

**ObjectKeyData:** These 8 bit values form an octet string that identifies the BIOP message that is the ServiceGateway message.

### B.2.8.2 DVB-J mounting of an object carousel

DVB-J causes an object carousel to be mounted using `ServiceDomain.attach()`. It can be unmounted using `ServiceDomain.detach()`.

An application manager is also allowed to call these methods implicitly when launching or killing an application in order to access the signalled base directory of the application.

### B.2.9 Unavailability of a carousel

Carousels become unavailable from the broadcast due to changes in the signalling including the following:

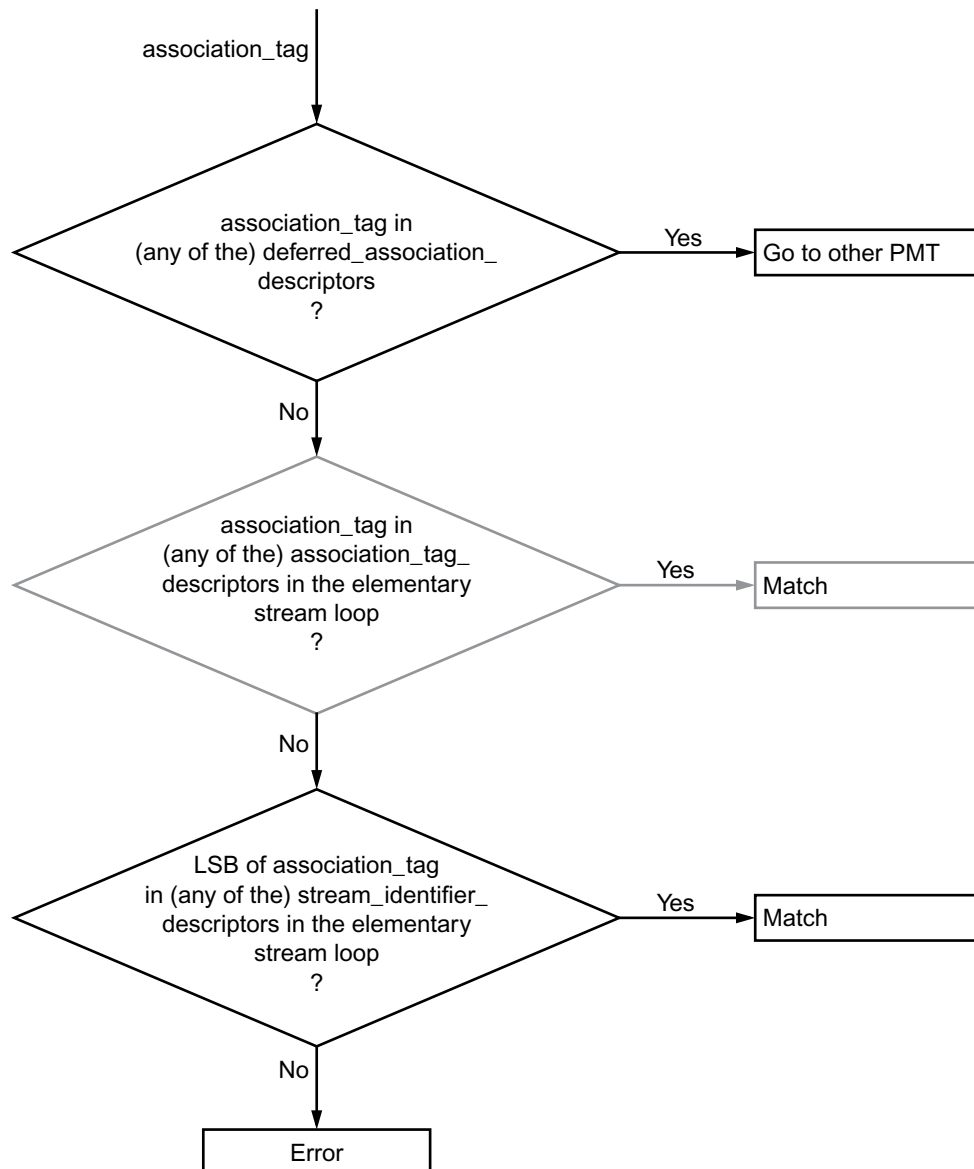
- The component signalled as carrying the DSI is removed from the PMT.
- Any value in the DSI changes.
- The value of carousel ID associated with the carousel changes.
- The program disappears from the PAT
- After an implementation dependent time general failure of the signalling (e.g. non-transmission of the PMT).

## B.3 AssociationTag Mapping

### B.3.1 Decision algorithm for association tag mapping

#### B.3.1.1 TapUse is **not** BIOP\_PROGRAM\_USE

The following figure illustrates the decision tree for identifying the elementary stream(s) by which the object carousel is distributed:



**Figure B.1 : Object Carousel ES identification decision tree**

In this specification, the stream\_identifier\_descriptor shall always be used for assigning a component\_tag for the elementary streams. Use of association\_tag\_descriptors is not required. If the association\_tag\_descriptor is optionally used, a stream\_identifier\_descriptor shall still be present and the tag values shall be set consistently in each descriptor. This restriction simplifies the decision tree above so that the second decision can be skipped.

### B.3.1.2 TapUse is BIOP\_PROGRAM\_USE

The decision tree in B.1 is not followed when resolving a BIOP\_PROGRAM\_USE tap as the only valid broadcast encoding is for a tap of use BIOP\_PROGRAM\_USE to resolve to deferred\_association\_tags\_descriptor in the PMT even if the deferred\_association\_tags\_descriptor identify the current service (i.e stream or streamEvent reference itself). If this resolution fails then there is an error in the broadcast.

### B.3.2 DSM-CC association\_tags to DVB component\_tags

The component\_tag in a PMT's stream\_identifier\_descriptor is used to relate SI service component information with an elementary stream without directly referring to a PID value. Likewise, association\_tags are used by DSM-CC in order to refer to an elementary stream without directly referencing a PID value. An association\_tag value is mapped to an elementary stream by matching the LSB of the association\_tag with a component\_tag. The stream\_identifier\_descriptor is mandatory for all components referenced by an application and/or object carousel.

Broadcasters may choose to use association\_tag\_descriptors (as defined by ISO/IEC 13818-6 [26]) which should (theoretically) be tested for a match before trying component\_tags. However, the LSB of the association\_tag value in an association\_tag\_descriptor has to be equal to the component\_tag for that PID. Since the component\_tag is unique within a PMT this removes the need to match against association\_tag\_descriptors.

The deferred\_association\_tag\_descriptor, as defined by ISO/IEC 13818-6 [26] section 11.4.3, is used to refer an association\_tag to a different PMT (i.e. a different service). When attempting to map an association\_tag to an elementary stream the association\_tag must first be checked against any deferred\_association\_tag\_descriptors in the current PMT (current in this context means the PMT of the service within which the association\_tag is being mapped). If the association\_tag matches any of the association\_tags present in a deferred\_association\_tag\_descriptor then the matching process restarts in the service indicated in that descriptor. If the transport\_stream\_id field in the deferred\_association\_tag\_descriptor is set to 0x0000 then it shall be ignored and the MHP terminal is free to choose which transport stream ID it selects when obtaining a service.

### B.3.3 deferred\_association\_tag\_descriptor

The transport\_stream\_id field may take value 0x0000 in which case it shall be ignored in resolving this reference.

## B.4 Example of an Object Carousel (informative)

The figure below illustrates an object carousel that is distributed over three elementary streams belonging to the same service.

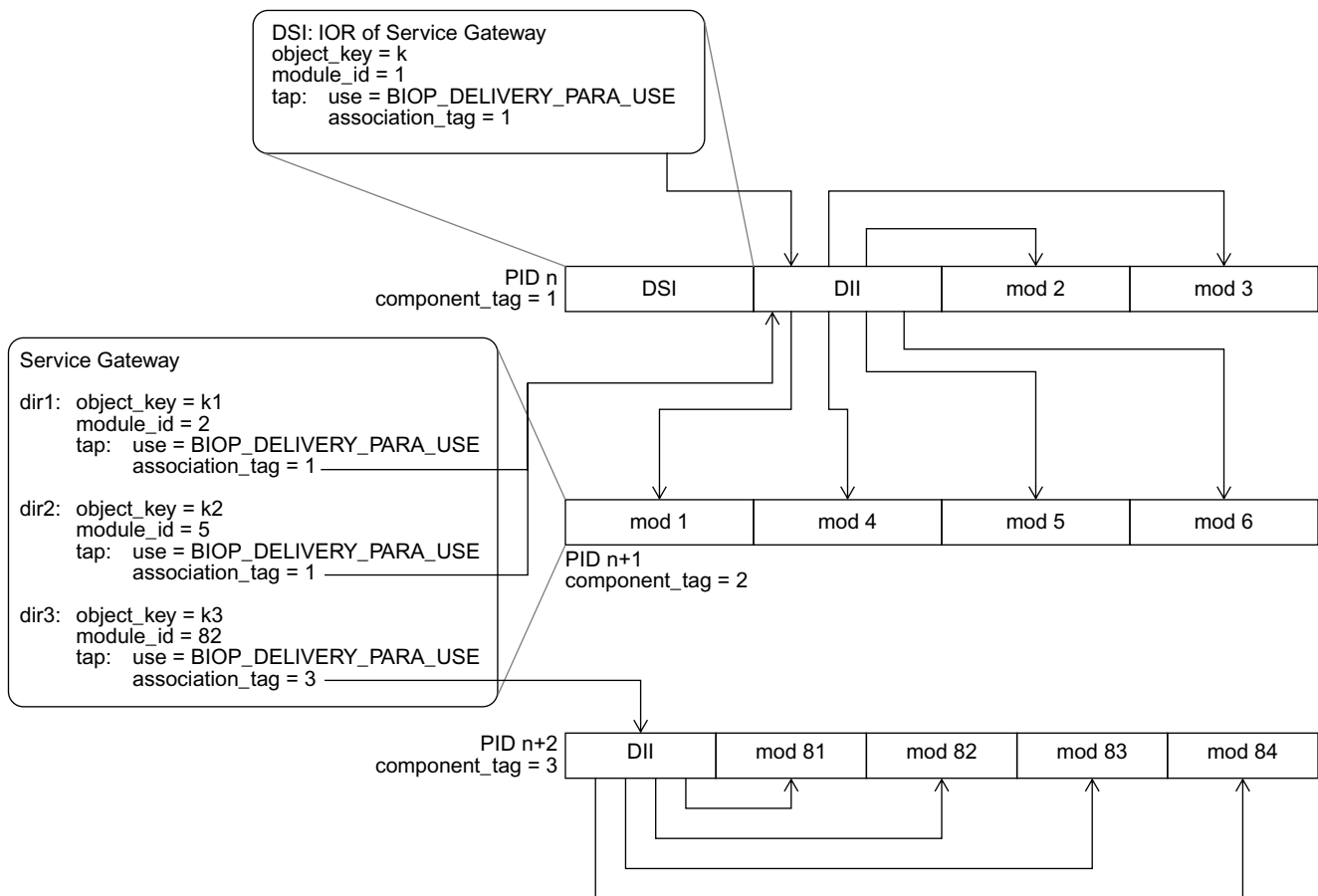


Figure B.2 : Example carousel

The DownloadServerInitiate (DSI) message is carried on the first elementary stream. It contains the object reference that points to the ServiceGateway. The tap with the BIOP\_DELIVERY\_PARA\_USE points to a DownloadInfoIndication (DII) message that provides the information about the module and the location where the module is being broadcasted. In the example, the ServiceGateway object is in the module number 1 that is carried on the second elementary stream (indicated by a BIOP\_OBJECT\_USE tap structure in the DII message).

The ServiceGateway object is a root directory that, in this example, references three subdirectories. Taps with BIOP\_DELIVERY\_PARA\_USE are used in the object references of the subdirectories to provide links to the modules via the DownloadInfoIndication (DII) message. The two first subdirectories "dir1" and "dir2" are referenced in the DII message that is carried in the first elementary stream. The third subdirectory is referenced in the DII message carried in the third elementary stream.

In this example, the two first elementary streams carry the messages of one logical data carousel while the third elementary stream carries the messages of another logical data carousel. All these belong to the same object carousel. In the example, the third elementary stream contains the objects in the "dir3" subdirectory and the objects in the "dir1" and "dir2" subdirectories are distributed over the first and second elementary stream.

It is important to note that the third elementary stream may originate from a completely separate source than the first two elementary streams. The directory hierarchy and objects contained in the third elementary stream are "mounted" in the root directory by providing the "dir3" directory entry with the appropriate location information.

This type of structure could be used, for example, in a national information service that contains some regional parts. The common national parts could be carried in this example case on the two first elementary streams that are distributed unmodified in the whole country. The regional parts are carried in the third elementary stream that is locally inserted at each region. From the application's point of view, the common national parts are in the "dir1" and "dir2" subdirectories while the regional parts are in the "dir3" subdirectory.

Another example where this type of structure could be used is if the service contains multiple independent applications. In this case, each application could be placed in its own subdirectory and these subdirectories might be carried as separate data carousels on different elementary streams.

## B.5 Caching

This section describes the constraints that an MHP terminal compliant with this specification shall implement when caching any content from the object carousel in the memory of the MHP terminal. Caching is optional for the MHP terminal, but if implemented shall conform to the constraints set in this section.

### B.5.1 Determining file version

There is no version number directly related to files (or other BIOP messages), the closest association is the moduleVersion in the DII that references the module that contains the BIOP message. Therefore, to ensure that a file is up to date the MHP terminal must determine that the moduleVersion for the appropriate module is current and reacquire if necessary.

When this checking is required is defined by the transparency level as specified in the following section.

### B.5.2 Transparency levels of caching

The definition of transparency levels describes the behaviour that the MHP terminal shall implement when the content in the object carousel is changing. The transparency level determines how certain the MHP terminal is required to be about the validity of the content when returning the content to the application. The object carousel provides a mechanism for determining version changes of the content by monitoring the DII messages.

Validity of content is specified here in terms of the version number of the module that is broadcast in the DII message. The contents of an object as cached in the memory of the MHP terminal are defined to be valid at a certain point in time when the version number of the module in the cache matches the version number of the module as signalled in the DII message describing that module as it was last broadcast. Note that the definition is based on the DII message that was last broadcast and it may be that the MHP terminal was not filtering for this message at that time and did not receive it.

From the MHP terminal point of view, the transparency level indicates the constraints that the terminal needs to implement for monitoring the DII messages.

The broadcaster can indicate the appropriate transparency level that shall be applied for a given piece of content by using a descriptor associated with a module in the DII message (see "[Caching priority descriptor](#)" on page 391). In the absence of this descriptor from a module, the transparent caching is the default level.

#### B.5.2.1 Transparent caching

The transparent caching is a caching level that ensures that the application can not practically notice a difference in the validity of the returned content between an implementation that caches content and an implementation that does not cache any content. Naturally, an implementation that caches the content will return it to the application faster.

When returning content from the cache to the application, the MHP terminal shall ensure that the version number of the cached content matches the version number indicated in the current DII message describing that module. Once a DII has been received it can be assumed that it is current at least for 500 ms and after that period until receiving the next instance of the relevant DII. If filtering for that DII has not resumed by the end of this period, the state of that DII is to be considered unknown until it is received again.

Therefore, terminals must not return transparently cached data if it has waited more than half a second between receiving the relevant DII and *starting to filter* for that DII again. If the terminal does not resume filtering within the 500ms grace period, it must download the relevant DII again when it wishes to use that DII to check cache validity.

The choice of 500 ms is based on the normal timing uncertainty in data delivery through the broadcast chain and is independent of the repetition rate of the DII messages.

#### B.5.2.1.1 Active caching

There are several ways the MHP terminal can organise its caching strategy. One possible strategy is so-called active caching. This means that the terminal has a dedicated section filter for each DII message it needs to monitor. Keeping that filter continuously filtering for the DII guarantees that the terminal will notice the update of a module as soon as it happens and can thus be aware of the validity of all the content it has cached.

However, in some cases the DII messages might be sent with a very high repetition rate that may cause a high processing load because the terminal needs to do some processing every DII message that it receives. The 500 ms grace period is designed to help this, as it allows the terminal to stop the section filter for 500 ms after receiving the DII message. This lessens the processing burden on the terminal as it only needs to process each DII message twice a second, even if it may be repeated on the transmission much more frequently.

#### B.5.2.1.2 Passive caching

With active caching, the terminal may need to have a dedicated section filter reserved for each DII message that it needs to monitor. This would effectively limit the amount of content that can be cached, possibly to a very small number. Therefore, the terminal may choose a so-called passive caching strategy. This means that the terminal does not even try to monitor for the DII messages continuously, but each time an application wants to retrieve an object, it at that time retrieves the current DII and checks if the cached content is still valid. Although, this strategy imposes a delay before returning the content to the application, this delay is usually significantly smaller than having to retrieve the content from the broadcast stream.

#### B.5.2.1.3 DII repetition rate

It should be noted that the description of active and passive caching are only informative here and terminal implementations can use any strategy fulfilling the normative constraints set above. However, broadcasters should set the repetition rate of the DIIs so that a terminal implementing the passive caching strategy will provide the expected benefits of caching over a terminal implementing no caching.

### B.5.2.2 Semi-transparent caching

The semi-transparent caching level allows the MHP terminal to cache the data and also return slightly out-dated data to the application. The benefit of this caching level is that it allows terminals to cache larger quantities of content with a reasonable resource usage while allowing the data to be returned usually immediately to the application. The semi-transparent caching level provides less guarantees about validity of the content, but does not cause the delay implied by the passive caching strategy with the transparent caching level.

When returning content from the cache to the application, the terminal shall ensure that the version number of the cached content matches the version number indicated in a valid DII message describing that module. Once a DII has been received it can be assumed to be valid at least for 30 s and after that period until receiving the next instance of the relevant DII. If filtering for that DII has not resumed by the end of this period, the state of that DII is to be considered unknown until it is received again.

Therefore, terminals must not return semi-transparently cached data if it has waited more than 30 seconds between receiving the relevant DII and starting to filter for that DII again. If the terminal does not resume filtering within the 30 s grace period, it must download the relevant DII again when it wishes to use that DII to check cache validity.

#### B.5.2.2.1 Implications for the terminal (informative)

Reasons for selecting the 30 s value for the grace period in the semi-transparent caching level are different from the reasons for the 500 ms grace period in the transparent level. The 30 s grace period in this level is intended e.g. to allow terminals to keep typically a valid copy of each DII by retrieving each DII in a round robin fashion using a single section filter. Naturally, whether this goal can be achieved, depends on the repetition rate of the DIIs and the amount of content that is cached. If this is not possible, the terminal might use the passive caching strategy with this transparency level as well. These strategies are only examples and the terminal may implement any strategy as long the normative constraints defined above are fulfilled (this includes implementing no caching as it is optional, as well as treating the semi-transparent level the same as the transparent level).

### B.5.2.3 Static caching

When using the static caching transparency level, the MHP terminal shall check the validity of the cached content from the version number in the DII message when it is used for the first time during the lifetime of an application instance. After the first usage time, the MHP terminal does not need to check the validity of the content during the lifetime of that application instance.

#### B.5.2.3.1 Implications for the broadcaster (informative)

This has the implication, that content with this transparency level is appropriate for very static content that is updated only rarely and where the possible update of the content does not need to be noticed by the application during the lifetime of one application instance.

#### B.5.2.3.2 Implications for the terminal (informative)

The MHP terminal, however, is allowed to update the contents of the statically cached files if it notices that they have been updated in the carousel as well as use any caching strategy as long as the normative constraint defined above are fulfilled (this includes implementing no caching as it is optional, as well as treating the static level the same as the semi-transparent and/or the transparent level).

## Annex C (informative): References

	Reference	Edition	Description	Note
[A]	MHP045	Rev.11	Digital Video Broadcasting (DVB); Commercial requirements	
[B]	UK MHEG Profile	1.05	Digital Terrestrial Television MHEG-5 Specification, U. K. DTG	
[C]	Compilers	ISBN: 0201100886	Compilers: Principles, Techniques, and Tools by Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman (Contributor); Addison-Wesley Pub Co	
[D]	Porter-Duff		T. Porter and T. Duff, "Compositing Digital Images", SIGGRAPH 84, 253-259.	
[E]	Java Media Player guide	1.03, Nov 6, 1997	Sun Microsystems Java Media Player guide, Java Media Players. Version. <a href="http://java.sun.com/products/java-media/jmf/forDevelopers/playerguide/index.html">http://java.sun.com/products/java-media/jmf/forDevelopers/playerguide/index.html</a> .	
[F]	Java VM2	ISBN: 0-201-43294-3	The Java Virtual Machine Specification (2nd edition), T. Lindholm and F. Yellin, Addison-Wesley.	
[G]	Java Class Libraries Vol. 1	ISBN 0-201-31002-3	The Java Class Libraries, Second Edition, Volume 1 by Patrick Chan, Rosanna Lee and Douglas Kramer.	
[H]	Java Class Libraries Vol. 2	ISBN 0-201-31003-1	The Java Class Libraries, Second Edition, Volume 2 by Patrick Chan and Rosanna Lee.	
[I]	E-Book	1.1	EACEM Technical Report Number TR-030, Baseline Digital Terrestrial TV Receiver Specification.	
[J]	RFC 2068	January 1997	Hypertext Transfer Protocol version 1.1	
[K]	RFC 2838	May 2000	Uniform Resource Identifiers for Television Broadcasts	
[L]	Accessibility	5 May 1999	W3C Recommendation: "Web Content Accessibility Guidelines 1.0" <a href="http://www.w3.org/TR/WAI-WEBCONTENT">http://www.w3.org/TR/WAI-WEBCONTENT</a>	



---

## Annex D (normative): Text presentation

### D.1 Scope

This section addresses the following topics:

- How downloaded fonts are associated with applications and accessed by them
- The DVB-J APIs that are used for presenting text and their behaviour

Two levels of interface are addressed:

- Simple string rendering as supported by `java.awt.Graphics.drawString`
- More complex text object rendering as supported by DVB Text Layout Manager as described in U, "(normative): Extended graphics APIs" on page 753.

Other parts of this specification that are related to this topic are:

- For character sets supported by implementations see E, "(normative): Character set" on page 434.
- For the font families, sizes, styles and weights supported by implementations see and the presentation of this to the API G.4, "Resident fonts and text rendering" on page 445.
- For the content formats used to deliver fonts see 7.4, "Downloadable Fonts" on page 67.

### D.2 Fonts

#### D.2.1 Embedded fonts

See G.4, "Resident fonts and text rendering" on page 445.

#### D.2.2 Downloaded fonts

##### D.2.2.1 Font technology

See 7.4, "Downloadable Fonts" on page 67.

## D.2.2.2 Font index files

### D.2.2.2.1 Format of file

The font index file provides a mapping between a font face name and a file containing the font data. The file syntax is defined by the XML DTD shown in table D.1.

**Table D.1 : Font index file syntax definition**

```

<!ELEMENT fontdirectory (font+)>
  <!-- a font definition -->

<!ELEMENT font (name,fontformat,filename,style*,size?)>
  <!-- filename of the font file.
  Because the font directory is per directory, this should
  not contain any directories, but just be a file in that
  directory -->

<!ELEMENT filename (#PCDATA)>
  <!-- font format, e.g. "PFR" -->

<!ELEMENT fontformat (#PCDATA)>
  <!-- symbolic name of the font -->

<!ELEMENT style (#PCDATA)>
  <!-- font style -->

<!ELEMENT name (#PCDATA)>

<!ELEMENT size EMPTY>
<!ATTLIST size
  min CDATA "0"
  max CDATA "maxint"
>

```

The `PublicLiteral` to be used for specifying this DTD in document type declarations of the XML files is:

```
"-//DVB//DTD Font Directory 1.0//EN"
```

and the URL for the `SystemLiteral` is:

```
"http://www.dvb.org/mhp/dtd/fontdirectory-1-0.dtd"
```

The Name used in the document type declaration shall be "fontdirectory".

#### D.2.2.2.2 Element semantics

**font:** There shall be one font element per font file included in the font directory.

**name :** Contains the font face name of the font (e.g. "Helvetica")

**fontformat :** The file format of the font. For the PFR format used in this specification, this shall be "PFR".

**filename:** Relative path to the font file. This is relative to the directory containing the font index file. The separator character for directories is "/". As this is a relative path, it shall not begin with a "/" character.

**style :** The style elements contain the names of the styles of the font that are contained in this font file. The possible values for this specification are "PLAIN", "BOLD", "ITALIC" and "BOLD\_ITALIC". There is one style element included per style contained in the indicated font file, except when all the usable styles of the font are in the same file in which case the style elements can be left out. When different styles of the font are contained in separate files, these are included in the directory as separate font entities with the same name but different style and filename.

**size:** Indicates the size range for which this font file can be used. The min. attribute contains the minimum size in points (default is "0"). The max attribute contains the maximum size in points or "maxint" if the maximum size is not limited (default is "maxint").

If all the usable sizes of the font are generated using the same font file, the size element can be left out. If there are separate files for different sizes, these are included in the directory as separate font entities with the same name and style but different size definition and filename.

### D.2.2.2.3 Example

**Table D.2 : Example index file**

```
<?xml version="1.0"?>

<!DOCTYPE fontdirectory PUBLIC "-//DVB//DTD Font Directory 1.0//EN"
"http://www.dvb.org/mhp/dtd/fontdirectory-1-0.dtd">
<fontdirectory>
  <font>
    <name>Tiresias</name>
    <fontformat>PFR</fontformat>
    <filename>tiresias.pfr</filename>
    <style>BOLD</style>
  </font>
  <font>
    <name>Broadcaster X Screen Font</name>
    <fontformat>PFR</fontformat>
    <filename>brxsf.pfr</filename>
  </font>
</fontdirectory>
```

## D.2.2.3 Name and location of font index files

### D.2.2.3.1 General

The specification of font paths and fonts by an application are private to that application, they are not available to other applications.

If the application creates a Font object with a *font face name* that is the same as one of the receiver's resident fonts then the specified font is used instead of the resident font with the same name. So, a downloaded font with *font face name* of "Tiresias" or "SansSerif" overrides any resident font with that name (see G.4, "Resident fonts and text rendering" on page 445).

### D.2.2.3.2 Name of file

The file name shall be:

```
"dvb.fontindex"
```

### D.2.2.3.3 Location

The font index file shall be placed in the base directory of the application.

The referenced font files can be on other file systems. However, the other file system must have already been mounted for the file loading to be successful. Font loading problems caused by referencing an unmounted file system are handled with the behaviour specified in java.AWT for an unavailable font.

## D.2.2.4 Specification of fonts at run time

### D.2.2.4.1 DVB-J

The implementation locates the font file using the parameters passed to the `java.awt.Font` constructor.

```
public Font(String name,
            int style,
            int size)
```

The font file is located by searching the directory for a file where the name element matches the name parameter of the Font constructor, a `style` element matches the style parameter and the `size` parameter is within the limitations in the size element.

It is font format specific how the font information for a given style and size is encoded in the font file.

## D.3 Text rendering

### D.3.1 Philosophy

This section describes "logical" rules that ensure text flows predictably on all receivers and defines some rendering requirements to ensure that a minimum acceptable level of text legibility is achieved.

No restriction is placed on the rendering technology used in a receiver provided that it achieves the deterministic text flow characteristics and the minimum rendering requirements described in this section.

### D.3.2 Low and high level rendering

Two levels of interface are addressed:

#### D.3.2.1 Low level rendering

Simple string rendering as supported by :

- `java.awt.Graphics.drawString`
- `java.awt.Graphics.drawChars`
- `java.awt.Graphics.drawBytes`

This is referred to as "low level" rendering implying that the application author has significant responsibilities for ensuring that the text is visible. This rendering obeys the normal AWT rules. For example, the author is responsible for placing individual words or lines of text on to a component.

#### D.3.2.2 High level rendering

More complex text object rendering as supported by :

- `org.dvb.DVBTextLayoutManager`

This is referred to as "high level" rendering implying that the application author may need less effort to ensure that the text is visible. For example, the author could use the text layout manager to handle flowing paragraphs of text into an `org.havi.ui.HText` object

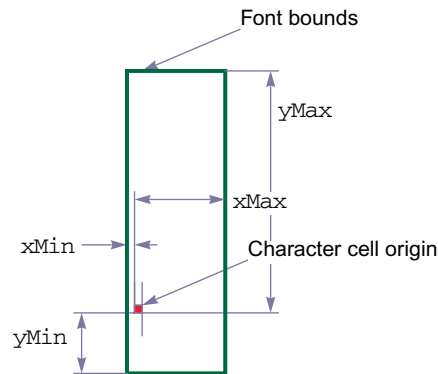
### D.3.3 Font Definition

The nomenclature used in this section is derived from the resident font format(s). The nomenclature and the numerics provided here are directly applicable to downloaded fonts.

Font bounds

The font definition induces a set of parameters `xMin`, `xMax`, `yMin` and `yMax` that are **properties of the font**. These define the maximum extent of the outline representation of characters within the physical font, and as such are defined in terms of outline resolution units (`outlineResolution`).

(xMin, yMin) and (xMax, yMax) are the bottom-left and top-right corners of an imaginary bounding rectangle within which all characters in the font can be completely enclosed.



**Figure D.1 : Font bounds**

In "Low level rendering" the author is responsible for using knowledge of these values to correctly position text. The following Java methods provide access to these parameters:

- `java.awt.FontMetrics.getMaxAscent` derives from `yMax`
- `java.awt.FontMetrics.getMaxDescent` derives from `yMin`
- `java.awt.FontMetrics.getMaxAdvance` derives from `xMax`

In each case the value returned is that in the font definition converted into the AWT device pixels. Typically there is a loss of precision in this process as the font is typically defined at higher resolution than the display device. In addition, precision may be lost if the aspect ratio of the font coordinate system is different from that of pixels in the display.

`java.awt.FontMetrics.stringWidth` contains the summated string widths of all of the characters in the string taking into account `xMin` & `xMax` plus any adjustments in the font such as kerning.

All approximations shall round up (see D.3.5, "Rendering within limits" on page 426).

In "High level rendering" the text layout manager uses this information when managing text flow to guarantee that the extremities of all characters are completely within the object. See D.3.5, "Rendering within limits" on page 426.

### D.3.3.1 "Physical" font data

"Physical" font data such as horizontal escapement and kerning is defined in terms of metrics resolution units (`metricsResolution`). This is a high resolution representation, abstracted from any actual rendering system.

NOTE: The `outlineResolution` and `metricsResolution` are not necessarily the same.

### D.3.4 Converting font metrics to display pixels

Many of the calculations in this section are in a high resolution physical coordinate system, either metrics or outline resolutions. These values need to be converted into the pixel resolution of the `HGraphicsDevice` to allow characters to be rendered.

Values in terms of these high level resolutions can be simply converted to values in terms of points by multiplying by the font size (in points) and dividing by the resolution, i.e. `metricsResolution` or `outlineResolution` as appropriate. However, this value in points still needs to be converted into a value in pixels.

Computer display systems typically assume a 72 pixel per inch display. So, as each point is 1/72 inch, the horizontal and vertical size of each pixel is 1 point.

### D.3.4.1 Vertical resolution

Each pixel in the graphics device containing the component is equivalent to a single point.

### D.3.4.2 Horizontal resolution

The horizontal relationship depends on the characteristics of the graphics device. For a square pixel graphics device the 1 pixel = 1 point convention can be preserved.

However, for a graphics device whose pixel aspect ratio is given by `org.havi.ui.HScreenConfiguration.getPixelAspectRatio` the horizontal resolution is the pixel aspect ratio \* 1 point.

**Table D.3 : Pixel width for non-square pixel graphics devices**

Graphics device resolution	Graphics device aspect ratio	Typographic pixel size width in points
720 x 576	4:3	128/117
	14:9	56/45
	16:9	512/351

An emulated graphics could be constructed with 14:9 aspect ratio. This could be used where text is required to be acceptable when viewed on either a 4:3 or 16:9 display. A possible example of this is illustrated in figure D.2.

Text on a 4:3 display

The quick brown fox jumped over the lazy dog.  
Cozy lummoX gives smart squid who asks for job pen.

Text on a 16:9 display

The quick brown fox jumped over the lazy dog.  
Cozy lummoX gives smart squid who asks for job pen.

**Figure D.2 : Example of 14:9 text on either 4:3 or 16:9 display**

### D.3.5 Rendering within limits

When typesetting for print, character extremities may extend beyond the nominal text flow area. However, print has margins so the edge of the text flow is not the technical limit to the area that can be printed. Component Insets can be used for the same purpose.

In "Low level rendering" the author is responsible for placing the text so that is not clipped.

In "High level rendering" the layout manager places the text within the inset rectangle, so that clipping does not occur if the object is sufficiently large. So, the "virtual margin" shall be defined as the greater of the inset value and the value computed using properties of the font (`xMin`, `yMin`, `xMax` and `yMax`) to ensure that all presented characters are completely rendered within the bounds of the object.

As stated previously, these parameters are defined in outline resolution units and so need to be converted to device pixels. Based on the principles described previously (see "Converting font metrics to display pixels") this can be achieved by using the following:

For `yMin` and `yMax`

$$Y_{\text{points}} = (\text{fontsize} \times Y_{\text{outlineResolution}}) / \text{outlineResolution}$$

$$Y_{\text{pixels}} = \text{roundupmag}(Y_{\text{points}})$$

and for  $x_{Min}$  and  $x_{Max}$

$$X_{points} = (\text{fontsize} \times X_{outlineResolution}) / \text{outlineResolution}$$

$$X_{pixels} = \text{roundupmag}(X_{points} / \text{pixel\_aspect\_ratio})$$

$\text{outlineResolution}$  is extracted from the font. The function  $\text{roundupmag}(A)$  rounds the magnitude of  $A$  to the first integral number greater than or equal to  $A$ , whilst preserving the sign, e.g.  $\text{roundupmag}(-4,32) = -5$ .

### D.3.5.1 Vertical limits

The origin of any character shall be at least  $y_{Max}$  inside the top edge of the object and at least  $y_{Min}$  inside its bottom edge. The distance from the edge of the component may be further increased by any margin. Assuming that all characters in a line of text share a common baseline then, **regardless of  $vAlign$**  the number of lines of text that may be presented within a text object is:

$$\text{num\_lines} = \text{floor}((\text{object\_height} - (y_{Min} + y_{Max})) / \text{linespace}) + 1$$

Where  $\text{object\_height}$  is the height of the component less any margin.

All values are in pixels. The variable  $\text{linespace}$  is an attribute of the object that defines the space between the baselines of consecutive lines of text. The function  $\text{floor}(A)$  rounds  $A$  to the first integral number less than or equal to  $A$ .

Note:  $\text{linespace}$  is defined in units of points but as described previously (see "[Converting font metrics to display pixels](#)").

- When  $vAlign = TOP\_ALIGN$  the baseline of the top most line shall be  $y_{Max}$  inside the "virtual margin" and each following line shall be spaced according to the value of  $\text{linespace}$ .

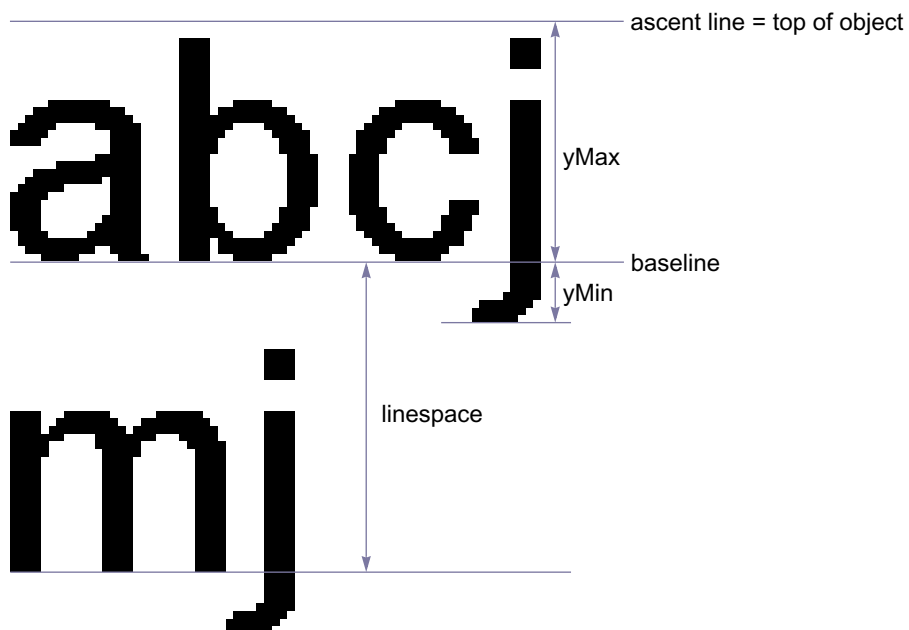


Figure D.3 : Vertical measures

- When  $vAlign = BOTTOM\_ALIGN$  the origin of the bottom most line shall be  $y_{Min}$  inside the "virtual margin" of the object and each previous line shall be spaced according to the value of  $\text{linespace}$ .
- For centring the midpoint of each line shall be the midpoint of the distance between the relevant "virtual margins"

### D.3.5.2 Horizontal limits

The number of characters that may be rendered on a line is not simply dependent upon the width of the object and the horizontal escapement for each character, but also needs to consider that the rendering of the first character in a line may extend to the left of its origin. Thus, **regardless of** `hAlign` the space available for rendering a line of text within an object is

$$\text{available\_width} = \text{object\_width} - \text{xMin}$$

All values are in pixels. `available_width` may then be used with the "logical" text width rules to determine text flow.

- When `hAlign = LEFT_ALIGN` the origin of the left most character shall be `xMin` inside the left edge of the "virtual margin".
- When `hAlign = RIGHT_ALIGN` the origin of the right most character shall be as necessary to ensure that it is completely visible when rendered.

### D.3.6 "logical" text width rules

This clause applies to both "[Low level rendering](#)" and "[High level rendering](#)". Its purpose is to ensure that text will flow predictably on different receivers and authoring stations, regardless of the quality of the character rendering, a set of "logical" text width rules are defined here.

NOTE: I.e. lines and words will break at the same character position.

These rules are a simplification of the rules that might be applied in a typographic rendering system. The objective of these simplifications is to reduce the receiver complexity required to ensure exact correlation of text flow behaviour.

The calculation of "logical" text width is based on "physical" font data. This data provides a description of the font at a very high resolution, abstracted from any actual rendering system. Consequently, the calculation of the "logical" width of a string of characters involves, computing their width at this high resolution and then converting to units appropriate to the rendering system, e.g. device pixels, before making decisions about text flow (see "[Converting font metrics to display pixels](#)").

In the case of "[Low level rendering](#)" it defines the internal computation performed by the AWT routines that measure the width of text:

- `java.awt.FontMetrics.charWidth`
- `java.awt.FontMetrics.stringWidth`
- `java.awt.FontMetrics.bytesWidth`

Due to the rounding processes within the calculations invoking these methods on subsets of a string may not yield the same total result as invoking the methods on the complete string. In particular the total of the values returned by `java.awt.FontMetrics.getWidths` may be different from the value returned by `java.awt.FontMetrics.stringWidth` for the same string.

In the case of "[High level rendering](#)" it defines the computations that the layout manager uses in the following cases:

- to determine when to wrap lines of text within an object
- to determine which tab stops text has passed when implementing tab characters

#### D.3.6.1 Computing "logical" text width

The key parameters when calculating the width of a string of N characters are:

- text font size
- `charSetWidth`
- The `metricsResolution`
- Any kerning adjustment



### D.3.6.1.1 Font sizes

Font sizes are expressed as the size of an "Em" in units of "points".

NOTE 1: Broadly speaking an Em is the minimum distance between the baselines of consecutive lines of text in the given font. If text is 48 point then the Em at that size is 48 points.

NOTE 2: The point is an archaic typographical unit. Traditionally there were 72,27 points to an inch. Computerised systems now use 72 points per inch for simplicity.

### D.3.6.1.2 Character widths

The font definition gives the width of each character relative to the size of an Em in metricsResolution units.

NOTE: If metrics are specified in 1/1000ths of an Em a character with a width of 0,6 Em will have a set width of 600.

### D.3.6.1.3 Kerning

For certain character combinations (a "kerning pair") a kerning adjustment may also be provided. Typically kerning reduces character spacing for pairs such as AV instead of A V, these provide a signed adjustment to the nominal charSetWidth of the first character.

Like charSetWidth kerning adjustments are in terms of metricsResolution units.

Kerning adjustments only apply between non whitespace characters, not between the start of a line of text and the edge of the text object. If justification is being used, only whitespace between words may be adjusted.

### D.3.6.1.4 Tracking

Tracking allows for an expansion/condensation of the character spacing for all of the characters in a text object including whitespace.

## D.3.6.2 Logical text width

The equation below shows how the width of a string of N characters is computed.

logical width of N characters<sub>points</sub> =

$$\text{div}((N - 1) \times \text{track}, 256) + \text{div}(\text{fontsize} \times \left( \sum_1^N \text{charSetWidth}[i] + \sum_1^{N-1} \text{kern}[i,i+1] \right), \text{metricsResolution})$$

$$\text{logical width of N characters}_{\text{pixels}} = \text{div}(\text{logical width of N characters}_{\text{points}} \times H, W)$$

Where in  $\text{div}(A, B)$ :

- B is unsigned and A is signed
- and
- $\text{div}(A, B) = \text{ceil}(A / B)$

Where '/' is a rational divide and  $\text{ceil}(A)$  is the first integral number greater than or equal to A. So, the calculations round up when reducing precision and tend to over estimate the width of text.

Where H and W are respectively the height and width returned by `org.havi.ui.HScreenConfiguration.getPixelAspectRatio()`.

## D.3.7 Line breaking

The text layout manager shall wrap text within the text object. The behaviour is equivalent to identifying (based on the "logical" length) the first word that won't fit completely within the text object and replacing the space character that precedes the word with a Carriage Return character. I.e. line breaks are only inserted where there are space characters, this implies the receiver does not have to apply word hyphenation rules.

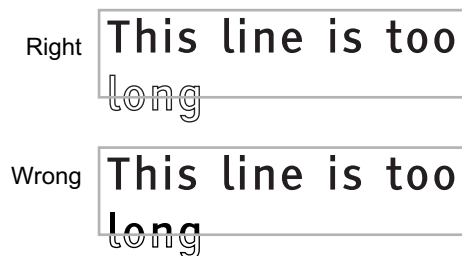
If a single word is bigger than the text object the text layout manager shall truncate text just before the last character that won't fit completely within the text box.

### D.3.7.1 Truncation of text

**High Level Text rendering shall not result in the presentation of partial characters.**

If an object is too small to display all of the text that it contains the text shall be truncated so that only complete characters are displayed, the `notifyTextOverflow` method will be called so that the application can alert the viewer or take other remedial action.

If the object is not tall enough to be able to display all of the lines of text loaded to it then the rendering shall only display lines whose height can be completely displayed.



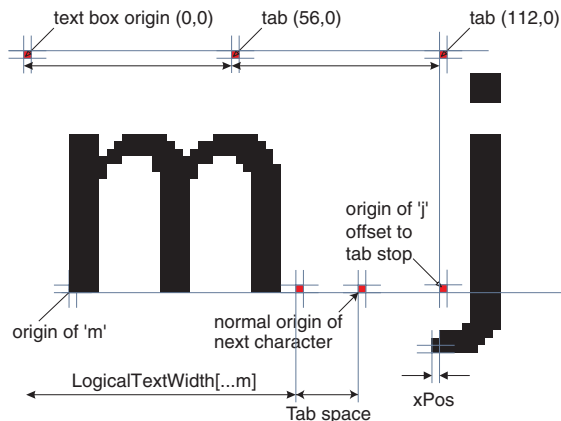
**Figure D.4 : Truncation at vertical "end" edge**

The examples illustrate the behaviour where `hAlign` is `LEFT_ALIGN` and `vAlign` is `TOP_ALIGN`. For other values the handling should be based on the same principles.

## D.3.8 Tabulation

In left aligned text tab stops are defined by default horizontally every 56 points from the left edge of the text box.

"Horizontal Tabulation" advances the origin of the next character to be rendered to the next tab stop in the direction that the text is currently flowing (the character repertoire in Table E.1 only requires left to right text).



**Figure D.5 : Effect of horizontal tabulation**

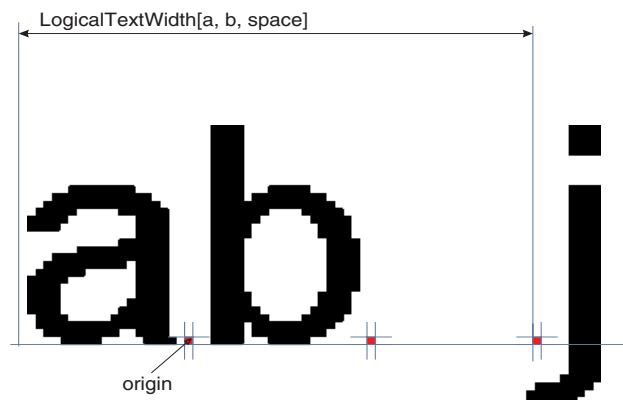
- Tab characters only have meaning in left aligned text. If the text is right aligned, centred or justified then tab character shall be treated as a space character.
- A tab logically advances the rendering of the text by at least the width of a space character. If the normal origin of the next character to be rendered after the tab character is after a tab stop, a tab character will advance the rendering to the subsequent tab stop.
- The tab stops are at regular intervals from the left edge of the object and are not affected by the `xMin` offset to the origin of the first character.

### D.3.9 Placing runs of characters & words

A run of characters starts from a well defined point:

- The start edge of the text object (see D.3.5 on page 426)
- A tab stop

After this origin the fine positioning of character cells and the gaps between words is not fully specified.



**Figure D.6 : Calculation of character placement**

However, the following rendering requirements shall be observed to ensure that a minimum acceptable level of text legibility is achieved:

- The spacing between any pair of non whitespace characters should be consistent wherever that pair of characters is displayed.
- At the default character spacing no two non-whitespace characters should "appear" to touch, except for special cases such as where ligatures diphthongs etc. are being used. The definition of the underlying font may make this requirement impossible.
- The physical rendering of a run of text as determined by the "logical" rules shall be achieved completely within the space used for the "logical" calculation.
- No partially rendered characters shall be presented.

### D.3.10 Control of text flow

See the definition of the DVB text layout manager.

## D.4 Text mark-up

This clause on text mark-up applies to "High level rendering".

### D.4.1 White Space Characters

Certain non-printing characters have special meaning. These are identified in [Table D.4](#).

**Table D.4 : Special characters**

UTF8 Value(s)	Name	Meaning
0x09	Tab	See "Tabulation" on page 430
0x0D	Carriage Return	Causes the text flow to break. The origin for the next character to be rendered moves to a new baseline "linespace" below that just rendered. The horizontal position of the next line will depend on the <code>hAlign</code> .
0x20	Space	Spaces text by the width defined for the space character. When a text object has the <code>TextWrapping</code> attribute set to "true" lines may be broken at a space. See <a href="#">D.3.7, "Line breaking" on page 430</a> .
0xC2 0xA0	Non-breaking space	Identical spacing characteristics to 0x20 but is not seen as word boundary for deciding a position to break a line of text. (0xC2A0 is the UTF-8 representation of 0x00A0)

### D.4.2 Marker characters

The codes 0x1C to 0x1F are zero width, non-spacing, non-printing characters available for use by authors as markers in text objects, i.e. when using string operations.

### D.4.3 Non-printing characters

Certain characters (or character sequences) have no immediate visual representation.

These include:

- 0x1C to 0x1F marker characters (see [D.4.2 on page 432](#))
- Format control mark-up (see [D.4.4 on page 433](#))
- other characters not recognised by the receiver

When presenting text that includes these characters the character placement shall be as if the non-printing characters were eliminated from the text before rendering. In particular, the character spacing and inter character kerning shall be computed as if the non-printing characters were not present.

#### D.4.4 Format Control Mark-up

Within text objects mark-up codes can be used to control the presentation of text. The sequence in table D.5 marks the start of some marked-up text. For each "start of mark-up" a corresponding "end of mark-up" is defined. The byte sequence for the "end of mark-up" is illustrated in Table D.6. The minimum number of supported mark-up instances, where each instance is a start and end mark-up pair, is 256.

**Table D.5 : General format for start of text mark-up**

	bits	value	note
start_of_markup	8	0x1B	Escape
markup_start_identifier	8	0x40-0x5E	"@" to "^"
parameters_length	8	N	
for( i=0; i<N; i++ ) { parameter_byte }	8	0x00...0xFF	

**Table D.6 : General format for end of text mark-up**

	bits	value	note
end_of_markup	8	0x1B	Escape
markup_end_identifier	8	0x60-0x7E	"" to "~"

**Table D.7 : Text object mark-up codes**

Min. Nesting	start mark-up	end mark-up	description
	0x1B 0x42 0x00	0x1B 0x62	Applies "bold" style to the text enclosed (note 1)
16	0x1B 0x43 0x04 0xrr 0xgg 0xbb 0xtt	0x1B 0x63	Applies colour to the text enclosed. 0xrr specifies the red intensity, 0xgg the green, 0xbb the blue and 0xtt the transparency.
NOTE 1: Not supported in this profile			

#### D.4.5 Future compatibility

Compatible extensions to the set of mark-up codes may be defined in future profiles. For each the markup\_end\_identifier will be 32 (0x20) greater than the markup\_start\_identifier. Engines shall ignore unrecognised mark-up and shall display any text enclosed within an unrecognised mark-up.

## Annex E (normative): Character set

### E.1 Basic Euro Latin character set

The MHP shall be able to *display* and *accept as input* at least the set of characters shown in table E.1 "Extended Latin set".

The range of characters accepted as input may be limited by the capabilities of the available input devices. This paragraph does not imply that keyboards (real and virtual) for MHP terminals are required to support input of this full character set.

**Table E.1 : Extended Latin set (Sheet 1 of 9)**

Code	Unicode 2.0 Character name
0x0020	SPACE
0x0021	EXCLAMATION MARK
0x0022	QUOTATION MARK
0x0023	NUMBER SIGN
0x0024	DOLLAR SIGN
0x0025	PERCENT SIGN
0x0026	AMPERSAND
0x0027	APOSTROPHE
0x0028	LEFT PARENTHESIS
0x0029	RIGHT PARENTHESIS
0x002A	ASTERISK
0x002B	PLUS SIGN
0x002C	COMMA
0x002D	HYPHEN-MINUS
0x002E	FULL STOP
0x002F	SOLIDUS
0x0030	DIGIT ZERO
0x0031	DIGIT ONE
0x0032	DIGIT TWO
0x0033	DIGIT THREE
0x0034	DIGIT FOUR
0x0035	DIGIT FIVE
0x0036	DIGIT SIX
0x0037	DIGIT SEVEN
0x0038	DIGIT EIGHT
0x0039	DIGIT NINE
0x003A	COLON
0x003B	SEMICOLON
0x003C	LESS-THAN SIGN
0x003D	EQUALS SIGN

**Table E.1 : Extended Latin set (Sheet 2 of 9)**

Code	Unicode 2.0 Character name
0x003E	GREATER-THAN SIGN
0x003F	QUESTION MARK
0x0040	COMMERCIAL AT
0x0041	LATIN CAPITAL LETTER A
0x0042	LATIN CAPITAL LETTER B
0x0043	LATIN CAPITAL LETTER C
0x0044	LATIN CAPITAL LETTER D
0x0045	LATIN CAPITAL LETTER E
0x0046	LATIN CAPITAL LETTER F
0x0047	LATIN CAPITAL LETTER G
0x0048	LATIN CAPITAL LETTER H
0x0049	LATIN CAPITAL LETTER I
0x004A	LATIN CAPITAL LETTER J
0x004B	LATIN CAPITAL LETTER K
0x004C	LATIN CAPITAL LETTER L
0x004D	LATIN CAPITAL LETTER M
0x004E	LATIN CAPITAL LETTER N
0x004F	LATIN CAPITAL LETTER O
0x0050	LATIN CAPITAL LETTER P
0x0051	LATIN CAPITAL LETTER Q
0x0052	LATIN CAPITAL LETTER R
0x0053	LATIN CAPITAL LETTER S
0x0054	LATIN CAPITAL LETTER T
0x0055	LATIN CAPITAL LETTER U
0x0056	LATIN CAPITAL LETTER V
0x0057	LATIN CAPITAL LETTER W
0x0058	LATIN CAPITAL LETTER X
0x0059	LATIN CAPITAL LETTER Y
0x005A	LATIN CAPITAL LETTER Z
0x005B	LEFT SQUARE BRACKET
0x005C	REVERSE SOLIDUS
0x005D	RIGHT SQUARE BRACKET
0x005F	LOW LINE
0x0061	LATIN SMALL LETTER A
0x0062	LATIN SMALL LETTER B
0x0063	LATIN SMALL LETTER C
0x0064	LATIN SMALL LETTER D
0x0065	LATIN SMALL LETTER E
0x0066	LATIN SMALL LETTER F
0x0067	LATIN SMALL LETTER G
0x0068	LATIN SMALL LETTER H
0x0069	LATIN SMALL LETTER I
0x006A	LATIN SMALL LETTER J
0x006B	LATIN SMALL LETTER K
0x006C	LATIN SMALL LETTER L

Table E.1 : Extended Latin set (Sheet 3 of 9)

Code	Unicode 2.0 Character name
0x006D	LATIN SMALL LETTER M
0x006E	LATIN SMALL LETTER N
0x006F	LATIN SMALL LETTER O
0x0070	LATIN SMALL LETTER P
0x0071	LATIN SMALL LETTER Q
0x0072	LATIN SMALL LETTER R
0x0073	LATIN SMALL LETTER S
0x0074	LATIN SMALL LETTER T
0x0075	LATIN SMALL LETTER U
0x0076	LATIN SMALL LETTER V
0x0077	LATIN SMALL LETTER W
0x0078	LATIN SMALL LETTER X
0x0079	LATIN SMALL LETTER Y
0x007A	LATIN SMALL LETTER Z
0x007B	LEFT CURLY BRACKET
0x007C	VERTICAL LINE
0x007D	RIGHT CURLY BRACKET
0x007E	TILDE
0x00A0	NO-BREAK SPACE
0x00A1	INVERTED EXCLAMATION MARK
0x00A3	POUND SIGN
0x00A5	YEN SIGN
0x00A9	COPYRIGHT SIGN
0x00AA	FEMININE ORDINAL INDICATOR
0x00AE	REGISTERED SIGN
0x00B0	DEGREE SIGN
0x00B7	MIDDLE DOT
0x00BA	MASCULINE ORDINAL INDICATOR
0x00BC	VULGAR FRACTION ONE QUARTER
0x00BD	VULGAR FRACTION ONE HALF
0x00BE	VULGAR FRACTION THREE QUARTERS
0x00BF	INVERTED QUESTION MARK
0x00C0	LATIN CAPITAL LETTER A WITH GRAVE
0x00C1	LATIN CAPITAL LETTER A WITH ACUTE
0x00C2	LATIN CAPITAL LETTER A WITH CIRCUMFLEX
0x00C3	LATIN CAPITAL LETTER A WITH TILDE
0x00C4	LATIN CAPITAL LETTER A WITH DIAERESIS
0x00C5	LATIN CAPITAL LETTER A WITH RING ABOVE
0x00C6	LATIN CAPITAL LETTER AE
0x00C7	LATIN CAPITAL LETTER C WITH CEDILLA
0x00C8	LATIN CAPITAL LETTER E WITH GRAVE
0x00C9	LATIN CAPITAL LETTER E WITH ACUTE

Table E.1 : Extended Latin set (Sheet 4 of 9)

Code	Unicode 2.0 Character name
0x00CA	LATIN CAPITAL LETTER E WITH CIRCUMFLEX
0x00CB	LATIN CAPITAL LETTER E WITH DIAERESIS
0x00CC	LATIN CAPITAL LETTER I WITH GRAVE
0x00CD	LATIN CAPITAL LETTER I WITH ACUTE
0x00CE	LATIN CAPITAL LETTER I WITH CIRCUMFLEX
0x00CF	LATIN CAPITAL LETTER I WITH DIAERESIS
0x00D0	LATIN CAPITAL LETTER ETH
0x00D1	LATIN CAPITAL LETTER N WITH TILDE
0x00D2	LATIN CAPITAL LETTER O WITH GRAVE
0x00D3	LATIN CAPITAL LETTER O WITH ACUTE
0x00D4	LATIN CAPITAL LETTER O WITH CIRCUMFLEX
0x00D5	LATIN CAPITAL LETTER O WITH TILDE
0x00D6	LATIN CAPITAL LETTER O WITH DIAERESIS
0x00D7	MULTIPLICATION SIGN
0x00D8	LATIN CAPITAL LETTER O WITH STROKE
0x00D9	LATIN CAPITAL LETTER U WITH GRAVE
0x00DA	LATIN CAPITAL LETTER U WITH ACUTE
0x00DB	LATIN CAPITAL LETTER U WITH CIRCUMFLEX
0x00DC	LATIN CAPITAL LETTER U WITH DIAERESIS
0x00DD	LATIN CAPITAL LETTER Y WITH ACUTE
0x00DE	LATIN CAPITAL LETTER THORN
0x00DF	LATIN SMALL LETTER SHARP S
0x00E0	LATIN SMALL LETTER A WITH GRAVE
0x00E1	LATIN SMALL LETTER A WITH ACUTE
0x00E2	LATIN SMALL LETTER A WITH CIRCUMFLEX
0x00E3	LATIN SMALL LETTER A WITH TILDE
0x00E4	LATIN SMALL LETTER A WITH DIAERESIS
0x00E5	LATIN SMALL LETTER A WITH RING ABOVE
0x00E6	LATIN SMALL LETTER AE
0x00E7	LATIN SMALL LETTER C WITH CEDILLA
0x00E8	LATIN SMALL LETTER E WITH GRAVE
0x00E9	LATIN SMALL LETTER E WITH ACUTE
0x00EA	LATIN SMALL LETTER E WITH CIRCUMFLEX
0x00EB	LATIN SMALL LETTER E WITH DIAERESIS
0x00EC	LATIN SMALL LETTER I WITH GRAVE

Table E.1 : Extended Latin set (Sheet 5 of 9)

Code	Unicode 2.0 Character name
0x00ED	LATIN SMALL LETTER I WITH ACUTE
0x00EE	LATIN SMALL LETTER I WITH CIRCUMFLEX
0x00EF	LATIN SMALL LETTER I WITH DIAERESIS
0x00F0	LATIN SMALL LETTER ETH
0x00F1	LATIN SMALL LETTER N WITH TILDE
0x00F2	LATIN SMALL LETTER O WITH GRAVE
0x00F3	LATIN SMALL LETTER O WITH ACUTE
0x00F4	LATIN SMALL LETTER O WITH CIRCUMFLEX
0x00F5	LATIN SMALL LETTER O WITH TILDE
0x00F6	LATIN SMALL LETTER O WITH DIAERESIS
0x00F7	DIVISION SIGN
0x00F8	LATIN SMALL LETTER O WITH STROKE
0x00F9	LATIN SMALL LETTER U WITH GRAVE
0x00FA	LATIN SMALL LETTER U WITH ACUTE
0x00FB	LATIN SMALL LETTER U WITH CIRCUMFLEX
0x00FC	LATIN SMALL LETTER U WITH DIAERESIS
0x00FD	LATIN SMALL LETTER Y WITH ACUTE
0x00FE	LATIN SMALL LETTER THORN
0x00FF	LATIN SMALL LETTER Y WITH DIAERESIS
0x0100	LATIN CAPITAL LETTER A WITH MACRON
0x0101	LATIN SMALL LETTER A WITH MACRON
0x0102	LATIN CAPITAL LETTER A WITH BREVE
0x0103	LATIN SMALL LETTER A WITH BREVE
0x0104	LATIN CAPITAL LETTER A WITH OGONEK
0x0105	LATIN SMALL LETTER A WITH OGONEK
0x0106	LATIN CAPITAL LETTER C WITH ACUTE
0x0107	LATIN SMALL LETTER C WITH ACUTE
0x0108	LATIN CAPITAL LETTER C WITH CIRCUMFLEX
0x0109	LATIN SMALL LETTER C WITH CIRCUMFLEX
0x010A	LATIN CAPITAL LETTER C WITH DOT ABOVE
0x010B	LATIN SMALL LETTER C WITH DOT ABOVE
0x010C	LATIN CAPITAL LETTER C WITH CARON
0x010D	LATIN SMALL LETTER C WITH CARON
0x010E	LATIN CAPITAL LETTER D WITH CARON
0x010F	LATIN SMALL LETTER D WITH CARON

Table E.1 : Extended Latin set (Sheet 6 of 9)

Code	Unicode 2.0 Character name
0x0110	LATIN CAPITAL LETTER D WITH STROKE
0x0111	LATIN SMALL LETTER D WITH STROKE
0x0112	LATIN CAPITAL LETTER E WITH MACRON
0x0113	LATIN SMALL LETTER E WITH MACRON
0x0116	LATIN CAPITAL LETTER E WITH DOT ABOVE
0x0117	LATIN SMALL LETTER E WITH DOT ABOVE
0x0118	LATIN CAPITAL LETTER E WITH OGONEK
0x0119	LATIN SMALL LETTER E WITH OGONEK
0x011A	LATIN CAPITAL LETTER E WITH CARON
0x011B	LATIN SMALL LETTER E WITH CARON
0x011C	LATIN CAPITAL LETTER G WITH CIRCUMFLEX
0x011D	LATIN SMALL LETTER G WITH CIRCUMFLEX
0x011E	LATIN CAPITAL LETTER G WITH BREVE
0x011F	LATIN SMALL LETTER G WITH BREVE
0x0120	LATIN CAPITAL LETTER G WITH DOT ABOVE
0x0121	LATIN SMALL LETTER G WITH DOT ABOVE
0x0122	LATIN CAPITAL LETTER G WITH CEDILLA
0x0123	LATIN SMALL LETTER G WITH CEDILLA
0x0124	LATIN CAPITAL LETTER H WITH CIRCUMFLEX
0x0125	LATIN SMALL LETTER H WITH CIRCUMFLEX
0x0126	LATIN CAPITAL LETTER H WITH STROKE
0x0127	LATIN SMALL LETTER H WITH STROKE
0x0128	LATIN CAPITAL LETTER I WITH TILDE
0x0129	LATIN SMALL LETTER I WITH TILDE
0x012A	LATIN CAPITAL LETTER I WITH MACRON
0x012B	LATIN SMALL LETTER I WITH MACRON
0x012E	LATIN CAPITAL LETTER I WITH OGONEK
0x012F	LATIN SMALL LETTER I WITH OGONEK
0x0130	LATIN CAPITAL LETTER I WITH DOT ABOVE
0x0131	LATIN SMALL LETTER DOTLESS I
0x0132	LATIN CAPITAL LIGATURE IJ
0x0133	LATIN SMALL LIGATURE IJ
0x0134	LATIN CAPITAL LETTER J WITH CIRCUMFLEX



Table E.1 : Extended Latin set (Sheet 7 of 9)

Code	Unicode 2.0 Character name
0x0135	LATIN SMALL LETTER J WITH CIRCUMFLEX
0x0136	LATIN CAPITAL LETTER K WITH CEDILLA
0x0137	LATIN SMALL LETTER K WITH CEDILLA
0x0138	LATIN SMALL LETTER KRA
0x0139	LATIN CAPITAL LETTER L WITH ACUTE
0x013A	LATIN SMALL LETTER L WITH ACUTE
0x013B	LATIN CAPITAL LETTER L WITH CEDILLA
0x013C	LATIN SMALL LETTER L WITH CEDILLA
0x013D	LATIN CAPITAL LETTER L WITH CARON
0x013E	LATIN SMALL LETTER L WITH CARON
0x013F	LATIN CAPITAL LETTER L WITH MIDDLE DOT
0x0140	LATIN SMALL LETTER L WITH MIDDLE DOT
0x0141	LATIN CAPITAL LETTER L WITH STROKE
0x0142	LATIN SMALL LETTER L WITH STROKE
0x0143	LATIN CAPITAL LETTER N WITH ACUTE
0x0144	LATIN SMALL LETTER N WITH ACUTE
0x0145	LATIN CAPITAL LETTER N WITH CEDILLA
0x0146	LATIN SMALL LETTER N WITH CEDILLA
0x0147	LATIN CAPITAL LETTER N WITH CARON
0x0148	LATIN SMALL LETTER N WITH CARON
0x014A	LATIN CAPITAL LETTER ENG
0x014B	LATIN SMALL LETTER ENG
0x014C	LATIN CAPITAL LETTER O WITH MACRON
0x014D	LATIN SMALL LETTER O WITH MACRON
0x0152	LATIN CAPITAL LIGATURE OE
0x0153	LATIN SMALL LIGATURE OE
0x0154	LATIN CAPITAL LETTER R WITH ACUTE
0x0155	LATIN SMALL LETTER R WITH ACUTE
0x0156	LATIN CAPITAL LETTER R WITH CEDILLA
0x0157	LATIN SMALL LETTER R WITH CEDILLA
0x0158	LATIN CAPITAL LETTER R WITH CARON
0x0159	LATIN SMALL LETTER R WITH CARON
0x015A	LATIN CAPITAL LETTER S WITH ACUTE
0x015B	LATIN SMALL LETTER S WITH ACUTE
0x015C	LATIN CAPITAL LETTER S WITH CIRCUMFLEX
0x015D	LATIN SMALL LETTER S WITH CIRCUMFLEX
0x015E	LATIN CAPITAL LETTER S WITH CEDILLA

Table E.1 : Extended Latin set (Sheet 8 of 9)

Code	Unicode 2.0 Character name
0x015F	LATIN SMALL LETTER S WITH CEDILLA
0x0160	LATIN CAPITAL LETTER S WITH CARON
0x0161	LATIN SMALL LETTER S WITH CARON
0x0162	LATIN CAPITAL LETTER T WITH CEDILLA
0x0163	LATIN SMALL LETTER T WITH CEDILLA
0x0164	LATIN CAPITAL LETTER T WITH CARON
0x0165	LATIN SMALL LETTER T WITH CARON
0x0166	LATIN CAPITAL LETTER T WITH STROKE
0x0167	LATIN SMALL LETTER T WITH STROKE
0x0168	LATIN CAPITAL LETTER U WITH TILDE
0x0169	LATIN SMALL LETTER U WITH TILDE
0x016A	LATIN CAPITAL LETTER U WITH MACRON
0x016B	LATIN SMALL LETTER U WITH MACRON
0x016C	LATIN CAPITAL LETTER U WITH BREVE
0x016D	LATIN SMALL LETTER U WITH BREVE
0x016E	LATIN CAPITAL LETTER U WITH RING ABOVE
0x016F	LATIN SMALL LETTER U WITH RING ABOVE
0x0172	LATIN CAPITAL LETTER U WITH OGONEK
0x0173	LATIN SMALL LETTER U WITH OGONEK
0x0174	LATIN CAPITAL LETTER W WITH CIRCUMFLEX
0x0175	LATIN SMALL LETTER W WITH CIRCUMFLEX
0x0176	LATIN CAPITAL LETTER Y WITH CIRCUMFLEX
0x0177	LATIN SMALL LETTER Y WITH CIRCUMFLEX
0x0178	LATIN CAPITAL LETTER Y WITH DIAERESIS
0x0179	LATIN CAPITAL LETTER Z WITH ACUTE
0x017A	LATIN SMALL LETTER Z WITH ACUTE
0x017B	LATIN CAPITAL LETTER Z WITH DOT ABOVE
0x017C	LATIN SMALL LETTER Z WITH DOT ABOVE
0x017D	LATIN CAPITAL LETTER Z WITH CARON
0x017E	LATIN SMALL LETTER Z WITH CARON
0x01CD	LATIN CAPITAL LETTER A WITH CARON
0x01CE	LATIN SMALL LETTER A WITH CARON
0x1E80	LATIN CAPITAL LETTER W WITH GRAVE
0x1E81	LATIN SMALL LETTER W WITH GRAVE
0x1E82	LATIN CAPITAL LETTER W WITH ACUTE
0x1E83	LATIN SMALL LETTER W WITH ACUTE

**Table E.1 : Extended Latin set (Sheet 9 of 9)**

<b>Code</b>	<b>Unicode 2.0 Character name</b>
0x1E84	LATIN CAPITAL LETTER W WITH DIAERESIS
0x1E85	LATIN SMALL LETTER W WITH DIAERESIS
0x1EF2	LATIN CAPITAL LETTER Y WITH GRAVE
0x1EF3	LATIN SMALL LETTER Y WITH GRAVE
0x2018	LEFT SINGLE QUOTATION MARK
0x2019	RIGHT SINGLE QUOTATION MARK
0x201C	LEFT DOUBLE QUOTATION MARK
0x201D	RIGHT DOUBLE QUOTATION MARK
0x2022	BULLET
0x2044	FRACTION SLASH
0x20AC	EURO SIGN
0x2190	LEFTWARDS ARROW
0x2191	UPWARDS ARROW
0x2192	RIGHTWARDS ARROW
0x2193	DOWNWARDS ARROW
0x221E	INFINITY
0x266B	BEAMED EIGHTH NOTES
0x2713	CHECK MARK
0x2717	BALLOT X

---

## Annex F (informative): Authoring & Implementation Guidelines

### F.1 Authoring Guidelines

- Authoring guidelines are needed to specify those methods, classes and interfaces which are intended for use by implementations of JMF players. These methods, classes and interfaces are not intended for use by applications except for this purpose and that should be made clear.
- Authoring guidelines are needed to make it clear that it is optional for JMF controls to have an associated java.awt component. This is in the JMF documentation but the phrasing implies this an exceptional case. In many MHP receivers, the presence of such a component would be the exceptional case. To be completed.

### F.2 Implementation Guidelines

To be completed.

### F.3 Authoring guidelines for DVB-J

< java stuff to go here >

### F.4 Authoring guidelines for DVB HTML

#### F.4.1 CSS2 Authoring guidelines

The selection of the CSS2 subset in this specification has been influenced by the requirements of the conformance clauses as defined by the W3C. As such, some style properties and selectors have been selected in order to respect this requirement, although they might not be fully adapted to the TV environment and its constraints. The objective of this section is to provide content authors with some guidelines on the potential impacts the use of some selectors or properties might have. Some of those guidelines are due to CPU or other limitations that may with generations of receivers at the time of the writing. This might evolve with the arrival of new generations, however content authors should be aware of the potential existence of different generations of receivers in the field.

##### F.4.1.1 Selectors

Content authors are advised that the use of contextual selectors - descendent, child, adjacent sibling, :first-child - as well as the attribute and the attribute value selector might cause the pattern matching process to be slow when used through the DOM APIs. For better matching efficiency, the type, id and class selectors should be favoured

Content authors are advised that the use of the :first-letter and :first-line pseudo-elements might cause the rendering to be slow. See also guidelines for content generation and the :before and :after selectors.

##### F.4.1.2 Properties

###### F.4.1.2.1 Generalities

On a general basis, use of the "auto" value for properties should be avoided and explicit values should be preferred wherever possible.

###### F.4.1.2.2 Visual Formatting Model

###### F.4.1.2.2.1 "display" :

Use of the "table-header-group", "table-row-group", "table-footer-group", "table-column" and "table-column-group" values of this property might cause the rendering to be slow. These values are meant to make elements behave like the table elements. Their functionality should be well understood before using them.

#### F.4.1.2.2.2 "float", "clear" :

Use of those properties might slow down the positioning operation. Also, for better content predictability and interoperability, absolute positioning should be favoured (see below).

#### F.4.1.2.2.3 "position" :

Use of the "fixed" value of this property might cause the associated rendering to be slow, especially in the case where boxes with different stack levels and with some level of translucency are used through the "z-index" property. On a general basis, for better content predictability and interoperability, absolute positioning should be favoured. Also, Content authors should not make any assumptions on the way the fixed positioning functionality interacts with scrolling.

#### F.4.1.2.2.4 Generated Content, automatic numbering, and lists

On a general level, content generation provides convenient functionality to the content author. However, this is at the cost of increased complexity in the receiver implementation. Content authors are then advised that the use of the associated selectors (:before and :after) and properties ("content", "counter-increment", "counter-reset", "quotes") might slow down the overall rendering.

#### F.4.1.2.2.5 "marker-offset", "display" (value "marker") :

Use of the list properties such as "list-style-type", "list-style-image", "list-style-position", and "list-style" should be favoured to markers because of the induced complexity of the latter, its relationships with content generation and consequently its potential slowness in the rendering.

#### F.4.1.2.3 Colours and Background

##### F.4.1.2.3.1 "background-attachment" :

Use of the "fixed" value of this property might cause the associated rendering to be slow. Content authors should not make any assumptions on the way the fixed positioning functionality interacts with scrolling.

Moreover, content authors should not make assumption on the presence of this functionality in a receiver since it is optional in CSS2.

#### F.4.1.2.4 Visual Effects

##### F.4.1.2.4.1 "overflow" :

Use of the "scroll" or "auto" values of this property might not cause the presence of scrolling mechanisms as used in the desktop environment. Due to ergonomic reasons, implementations might choose other mechanisms.

#### F.4.1.2.5 Fonts

##### F.4.1.2.5.1 "font" :

Content authors should not make any assumption on the expected results when using the "caption", "icon", "menu", "message-box", "small-caption", "status-bar" values of the "font" property, which have no associated semantics in a DVB-MHP environment.

#### F.4.1.2.6 Text

##### F.4.1.2.6.1 "text-align" :

Use of the "justify" value of this property might slow down the rendering because of the induced complexity. Depending on the algorithm used by the receiver, use of this value might also affect interoperability. Content authors should be aware that conformant receivers might interpret this value as being "left" or "right" depending on the directionality of the language.

Use of a string value for this property in a table cell might also slow down the rendering and in general might be unpredictable, for example when the content flows on more than one line.

#### F.4.1.2.6.2 "text-shadow" :

Use of the optional blur radius after the shadow offset might slow down the rendering because of the induced complexity and may not be visible on a low resolution screen. Content authors should be advised that interoperability might be affected since the algorithm for computing the blur effect has not been specified.

#### F.4.1.2.7 User Interface

##### F.4.1.2.7.1 User preferences for colours :

Content authors should not make any assumption on the expected results when using the system colours which have no associated semantics in a DVB-MHP environment. Those colours are: ActiveBorder, ActiveCaption, AppWorkspace, Background, ButtonFace, ButtonHighlight, ButtonShadow, ButtonText, CaptionText, GrayText, Highlight, HighlightText, InactiveBorder, InactiveCaption, InactiveCaptionText, InfoBackground, InfoText, Menu, MenuText, Scrollbar, ThreeDDarkShadow, ThreeDFace, ThreeDHighlight, ThreeDLightShadow, ThreeDShadow, Window, WindowFrame, WindowText.

## Annex G (normative): Minimum Platform Capabilities

### G.1 Graphics

In the area of graphics capability the following requirements are made on MHP terminals:

#### G.1.1 Device capabilities

- The number of applications concurrently using the display is not limited. However, the MHP terminal is not required to support overlapping HScenes.
- The MHP terminal shall implement at least one HGraphicsDevice which shall be full screen.
- The MHP terminal shall implement at least one HBackgroundDevice. These are always full screen.
- The MHP terminal shall implement at least one HVideoDevice which is always capable of being configured to be full screen.
- The minimum set of required device resolutions that MHP terminals shall support is illustrated in figure G.1. Specifically these are:
  - HBackgroundDevice resolution of 720x576
  - HVideoDevice resolution of 720x576
  - HGraphicsDevice resolution of 720x576
 These shall be supported for display aspect ratios of 4:3 and 16:9.

Optionally MHP terminals may also support square pixel HGraphicsDevice resolutions of 768x576 and 1024x576 for 4:3 and 16:9 displays respectively.

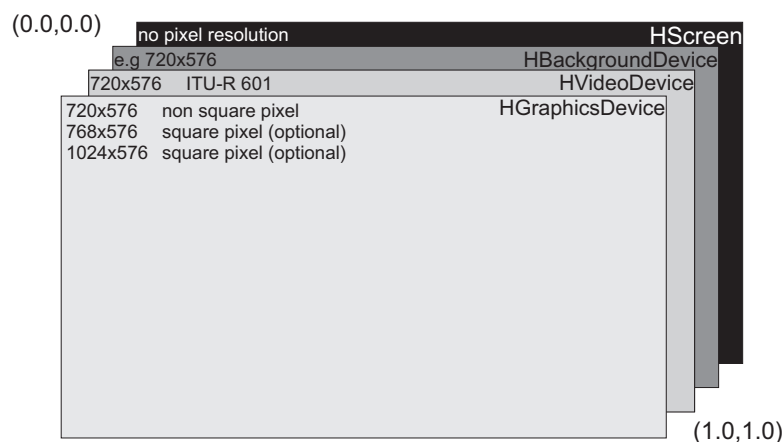


Figure G.1 : Required device resolutions

#### G.1.2 Video presentation capabilities

- The following set of standard decoder format conversions shall be supported by all MHP terminals:

DFC\_PROCESSING\_CCO

DFC\_PROCESSING\_FULL

DFC\_PROCESSING\_LB\_16\_9

DFC\_PROCESSING\_PAN\_SCAN

The following modes are optional:

DFC\_PROCESSING\_LB\_14\_9

DFC\_PROCESSING\_LB\_2\_21\_1\_ON\_16\_9

## DFC\_PROCESSING\_LB\_2\_21\_1\_ON\_4\_3

- MHP terminals are required to support both displaying MPEG video without any scaling and with 1/2 scaling both vertically and horizontally provided that in this latter case the entire resulting video area is fully on the screen
- Support for component based JMF players is not required for any profile in this specification.

### G.1.3 Image processing capabilities

- All DVGraphics objects shall support SRC and CLEAR and SRC\_OVER. When SRC\_OVER is used with DVGraphics objects with a sample model of the type TYPE\_BASE a perfect result is only guaranteed to be produced with alpha values of 0 and 1. Alpha values other than 0 and 1 can be approximated. DVGraphics object with a type of TYPE\_ADVANCED will produce a result as expected but those SRC\_OVER operations are likely to be slow.
- DVGraphics object created from a DVBufferedImage with the type TYPE\_ADVANCED shall perform SRC\_OVER operations without approximations of the compositing rule.

### G.1.4 Alpha capabilities

For any draw operations directly into the HGraphicsDevice the following rules shall be applied for the precision of implementation of alpha:

- MHP terminals are required to implement at least 3 levels of transparency: 0 % (opaque), 100 % (completely transparent) and an intermediate value of approximately 30 %. Implementation of additional intermediate levels of transparency is optional.
- Where the MHP terminal cannot implement a particular value of semi-transparency it shall replace it with the nearest value of transparency it can implement.

However, if the encoded value of transparency is in the range 10 %-90 % it shall not be approximated as either less than 10 % or greater than 90 % transparency.

So, 9 % may be approximated as 0 % but 10 % shall be represented with a value in the range 10 % to 90 % such as 30 %. Similarly, 91 % may be approximated as 100 %.

### G.1.5 Colour capabilities

Logically the colour model is a "true colour" one. However, other implementations are possible.

Two styles of indexed colour implementations are considered:

- Dithering
- Nearest colour match

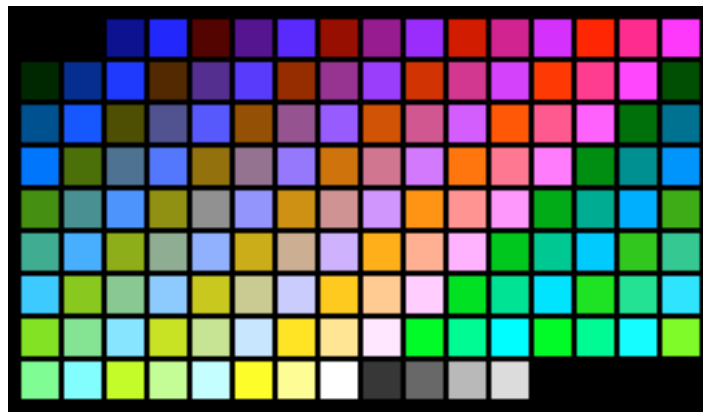
Where an indexed colour implementation can accurately reproduce colours using dithering it is considered to be a true colour implementation. In this case no restrictions are placed on the CLUT used.

Where an indexed colour receiver implements a simpler colour matching, or has other limitations on the number of colours it represents (for example requiring a reservation to accommodate subtitles). It shall use the 188 colour CLUT specified in table G.1. The reservation of 64 CLUT locations for use by the subtitling decoder is not appropriate to all implementations and assumes a corresponding broadcaster rule of operations restricting subtitle transmissions to use only 64 different colours.

**Table G.1 : Palette construction rules**

Transparency	Alpha	Additional Grey Levels (R=G=B)	Red	Green	Blue	Number of colours
0 %	255	42, 85, 170, 212	0, 63, 127, 191, 255	0, 31, 63, 95, 127, 159, 191, 223, 255	0, 127, 255	139
30 %	179 (note 1)	--	0, 85, 170, 255	0, 51, 102, 153, 204, 255	0, 255	48
100 %	0	--	--	--	--	1
					Total	188
NOTE 1: Where the receiver cannot implement this "ideal" value of semi-transparency it shall replace it with the nearest value of semi-transparency it can implement. Note: semi-transparency shall not be approximated as either 0 % or 100 % transparency.						

The opaque portion CLUT specified in table G.1 is illustrated in figure G.2.



**Figure G.2 : Opaque CLUT**

## G.1.6 MPEG I frame and Video drips

The minimum positioning and scaling capabilities defined above for MPEG video shall also apply to MPEG I frame and Video drips.

## G.2 Audio

No audio mixing is required.

Audio played from memory may pre-empt any audio from the transport stream. This may disturb decoding of any broadcast video stream.

Audio from memory shall be output in preference to audio from stream if overall audio output has not been disabled by the user. On platforms capable of mixing audio from memory with audio from the stream it shall do this if there is a stream playing. Where audio from the stream is interrupted, decoding of it shall automatically resume when audio from memory ceases if the stream concerned is still playing. This only applies where the audio from memory and the stream are both under the control of the same MHP application. Where multiple applications are involved see 9.5, "Inter application resource management" on page 202.



## G.3 Video

The MHP terminal is only required to support decoding of a single video stream at a given time. The number of implemented video decoders will affect the functionality of the video and background devices.

## G.4 Resident fonts and text rendering

### G.4.1 The built-in font

At least the RNIB/DTG font "Tiresias" shall be provided.

The font shall be able to present at least the sizes listed in G.2 and the weight ("PLAIN").

**Table G.2 : Minimum set of sizes**

Size (points)	TV lines (note 1) over "Cap-V"	Informative Name
36	24	Heading / Large subtitle
31	21	Subtitle
26	18	Body (note 2)
24	16	Footnote
NOTE 1: The primary definition of the character size is the font size in points, the height of a capital letter "V" in TV lines is provided for information only.		
NOTE 2: The default size and style.		

### G.4.2 Presentation to DVB-J

The embedded font "Tiresias" shall have:

- the logical name "SansSerif" (for example returned by `java.awt.Toolkit.getFontList`)
- the family name "Tiresias" (for example returned by `java.awt.Font.getFamily`)
- the font face name "Tiresias PLAIN"

### G.4.3 Text directions

The `DVBTextLayoutManager` is only required to support the following configuration of text direction:

- `LINE_ORIENTATION_HORIZONTAL` and `START_CORNER_UPPER_LEFT`

## G.5 Input events

Table G.3 : Minimum set of input events

Input event
VK_0 to VK_9
VK_UP
VK_DOWN
VK_LEFT
VK_RIGHT
VK_ENTER
VK_TELETEXT
VK_COLORED_KEY_0
VK_COLORED_KEY_1
VK_COLORED_KEY_2
VK_COLORED_KEY_3

## G.6 Memory

In order to be able to execute MHP conformance tests, the following minimum memory requirements are defined for MHP terminals. All of these are to be measured during normal usage and operational conditions of an MHP terminal. All are to be measured in the `initXlet` method of a DVB-J application which is both the only auto-start application signalled in a service and the only application running at that time.

- Enough memory to successfully load any arbitrary 262 144 (or less) Bytes of Java class files into the memory space of the Java virtual machine. Execution of code called as part of initializing fields in classes is excluded from consideration as part of "load"ing here. RAM usage by the bytecode verifier is included in consideration as part of "load"ing here.

Enough memory to do the above and individually each of the following:

- Enough memory to successfully create a Java byte array of lengths from 1 entry to 262 144 entries.
- Enough memory to successfully load & display any 720x576 8bit PNG image (conforming to 15.1, "PNG - restrictions" on page 363) from a file which contains just the mandatory information and excludes any optional extension fields or chunks.
- Enough memory to successfully load from file & play from memory 5 seconds of audio at 128 kbit/s (where kbit/s is as used in TR 101 154 [9]). In a similar way to as is required for PNG images above, it shall be measured using files that do not include any optional extension fields.
- Enough memory to successfully allocate an array of `java.lang.Object` of length 16 384 and fill each element of this array with a distinct instance of `java.lang.Object`.

The memory requirements detailed in this section are not exhaustive. For example, the specific requirement concerning an array of type `byte` in no way implies that MHP terminals are exempt from requirements found elsewhere in the MHP specification (including normatively referenced specifications) for supporting arrays of other types.

NOTE: Additional detail may be added to these requirements in order to properly enable the MHP conformance tests.

## G.7 Other resources

**Table G.4 : Minimum requirements for other resources**

Feature	Specification
gamma correction in the receiver	none
HAVi mattes	Platforms are not required to implement the functionality of mattes in HAVi. Non-implementation should be implemented as specified by HAVi.
Overlapping applications	MHP terminals are not required to support overlapping top level UI containers (e.g. HScenes where DVB-J applications are concerned).
AIT section filtering	The implementation is not required to dedicate more than one section filter to monitoring the AIT.
Key lengths	Receivers shall support certificate key length up to 4096 bits.
Internet clients	WWW and email clients required, usenet news client optional.
Non-CA Smart Card Reader	MHP terminals are not required to include this
Stored services	No requirement for persistent storage for stored services even if other forms of persistent storage are supported.

NOTE: The values in the table below are set for the purposes of conformance testing and should not be used by application or MHP terminal developers as being indicative of the capabilities of commercial products.

**Table G.5 : Minimum requirements for other resources for conformance purposes (Sheet 1 of 2)**

Feature	Specification
Application accessible timers (note 2)	At least 4 timers for each ServiceContext which can be presenting MHP applications at the same time. (i.e. shared between the applications signalled as part of the same service). (note 1)
MPEG-2 transport stream network interface	Shall support at least one network interface enabling reception of an MPEG-2 transport stream and selection of that transport stream from among those available to be received. This shall support those broadcast channel protocols supported by the MHP terminal (see 6.2, " <a href="#">Broadcast Channel Protocols</a> " on page 54) and the MHP APIs defined to interface to these protocols and to control these interfaces.
Bidirectional IP network interface	MHP terminals supporting the interactive broadcast profile shall support at least one network interface for bidirectional IP traffic. This shall support those interaction channel protocols supported by the MHP terminal (see 6.3, " <a href="#">Interaction Channel Protocols</a> " on page 57) and the MHP APIs defined to interface to these protocols and to control these interfaces.
Conditional access	None required. The absence or optional presence of one shall be correctly reported through the Conditional access API. Local regulation may require more support than this minimum
Persistent storage	No requirement for storage which persists across interruption of power to an MHP.
Application accessible MPEG-2 section filters (note 2)	At least 2 shared among all applications signalled as part of the same service. (note 1).
Application accessible DVB-J threads (note 2)	At least 4 shared among all applications signalled as part of the same service. Threads created by the platform and used to call methods of the application are excluded from this number. (note 1).

**Table G.5 : Minimum requirements for other resources for conformance purposes (Sheet 2 of 2)**

<b>Feature</b>	<b>Specification</b>
NOTE 1:	These requirements apply to one set of MHP applications signalled as part of the same service. If an MHP terminal supports simultaneous execution of more than one set of signalled applications then it shall make available at least these minimum resources for each set of signalled applications which can be executed simultaneously.
NOTE 2:	"Application accessible" means guaranteed to be accessible to MHP applications through the API defined in this specification for the feature concerned. This must be regardless of the extent of any usage of that feature or function as part of the MHP terminal implementation.

---

## **Annex H (normative): DVB-J fundamental classes**

# Package org.dvb.lang

## Description

Provides those core platform related features not found in the java.lang package.

### Class Summary

#### Classes

[DVBClassLoader](#)

This class loader is used to load classes and resources from a search path of URLs referring to locations where Java class files may be stored.

# org.dvb.lang DVBClassLoader

## Syntax

```
public class DVBClassLoader extends java.security.SecureClassLoader
```

```
java.lang.Object
|
+--java.lang.ClassLoader
|
+--java.security.SecureClassLoader
|
+--org.dvb.lang.DVBClassLoader
```

## Description

This class loader is used to load classes and resources from a search path of URLs referring to locations where Java class files may be stored.

The classes that are loaded are by default only allowed to load code through the parent classloader, or from the URLs specified when the DVBClassLoader was created.

## Constructors

---

### DVBClassLoader(URL[])

```
public DVBClassLoader(java.net.URL[] URLs)
```

Constructs a new DVBClassLoader for the given URLs. The URLs will be searched in the order specified for classes and resources.

If there is a security manager, this method first calls the security manager's `checkCreateClassLoader` method to ensure creation of a class loader is allowed.

#### Parameters:

URLs - the URLs from which to load classes and resources

#### Throws:

`SecurityException` - if a security manager exists and its `checkCreateClassLoader` method doesn't allow creation of a class loader.

#### See Also:

`SecurityManager`

---

### DVBClassLoader(URL[], ClassLoader)

```
public DVBClassLoader(java.net.URL[] URLs, java.lang.ClassLoader parent)
```

Constructs a new DVBClassLoader for the given URLs. The URLs will be searched in the order specified for classes and resources.

If there is a security manager, this method first calls the security manager's `checkCreateClassLoader` method to ensure creation of a class loader is allowed.

#### Parameters:

URLs - the URLs from which to load classes and resources

parent - the parent classloader for delegation

**Throws:**

`SecurityException` - if a security manager exists and its `checkCreateClassLoader` method doesn't allow creation of a class loader.

**See Also:**

`SecurityManager`

## Methods

---

**findClass(String)**

```
public java.lang.Class findClass(java.lang.String name)
```

Finds and loads the class with the specified name from the URL search path. Any URLs are searched until the class is found.

**Parameters:**

`name` - the name of the class.

**Returns:**

the resulting class.

**Throws:**

`ClassNotFoundException` - if the named class could not be found.

---

**newInstance(URL[])**

```
public static DVBClassLoader newInstance(java.net.URL[] URLs)
```

Creates a new instance of `DVBClassLoader` for the specified URLs. If a security manager is installed, the `loadClass` method of the `DVBClassLoader` returned by this method will invoke the `SecurityManager.checkPackageAccess` method before loading the class.

**Parameters:**

`URLs` - the URLs to search for classes and resources.

**Returns:**

the resulting class loader

---

**newInstance(URL[], ClassLoader)**

```
public static DVBClassLoader newInstance(java.net.URL[] URLs, java.lang.ClassLoader parent)
```

Creates a new instance of `DVBClassLoader` for the specified URLs. If a security manager is installed, the `loadClass` method of the `DVBClassLoader` returned by this method will invoke the `SecurityManager.checkPackageAccess` method before loading the class.

**Parameters:**

`URLs` - the URLs to search for classes and resources.

`parent` - the parent class loader for delegation.

**Returns:**

the resulting class loader



## Annex I (normative): DVB-J event API

Applications can use the `org.dvb.event` API, to receive events without being focused and/or to have exclusive access to events.

### I.1 Overview

This API provides a mechanism allowing MHP applications to influence the routing of events to either MHP applications or the navigator. This typically would be used in a mode where the receiver is primarily used for TV viewing, allowing some events to be received by the application, and allowing the navigator to receive those events that are not requested by any MHP application. This API enables MHP applications to choose between the following mechanisms for receiving events:

- through the standard `java.awt` mechanism,
- through the standard `java.awt` mechanism, but for some events to be exclusively accessed by the application,
- through a mechanism defined by this API,
- through the mechanism defined by this API, but for some events to be exclusively accessed by the application.

The last two solutions could be used by non-graphical applications in order to receive events that are coming from the user. It could also be used by an invisible application if it wants to be presented when a specific key is pressed.

If an application wants to have exclusive access to some events and to manage them through the `java.awt` then it must use this API so that it can be aware of the fact that it has lost or gained access to these events. One must notice that an application based on `awt` event mechanism will receive events only if it is focused.

The diagram below shows how an event is dispatched depending on how it was requested:

```

User Event
|
|
| . Is it in a repository used by the Event Manager
| or is there an application with AWT focus
| |
yes  no
|   |
|   |--> send the event to the navigator
|
| . Is it in a repository used by the Event Manager?
| |
yes  no
|   |
|   |--> send the event to the awt application which is focused.
|
| . Is it an exclusive event?
| |
yes  no
|   |
|   |--> send the event to the applications which do not use the awt
|           event mechanism and have asked for this event.
|
| . Was it acquired by an awt application?
| |
yes  no

```

```

|   |
|   .-> send the event to the application which does not use the
|       awt event mechanism and has asked for an exclusive access
|       to this event.
|
|
. ----> send the event to the application if it is focused.

```

## 1.2 The resource management

An application asking for exclusive access to some events will use the resource framework defined in [DAVIC 1.4.1p9 \[3\]](#) so that it can be aware of the fact that it has lost access to the user events it asked for (see the example below).

## 1.3 The Event Repository

The `UserEventRepository` is the class that is used by the application to define the user events it intends to use. For the moment user events are just key events but it is a place-holder for new families of events (voice command for example). If an application asks for an exclusive access to events by means of a repository, this exclusive access will be lost at the time when one of the event is grabbed by another application. User events that can be accessed by an application are defined in the `UserEvent` class.

### 1.3.1 Example

exclusive access to events for a non-focused application

```

import org.davic.resources.ResourceClient.* ;
import org.dvb.event.* ;
class Example implements UserEventListener, ResourceStatusListener, ResourceClient {
    private int myStatus ;
    public Example () {
        EventManager em ;
        UserEventRepository repository ;
        em = EventManager.getInstance () ;
        repository = new UserEventRepository ("R1") ;
        repository.addKey (UserEvent.VK_ENTER) ;
        em.addUserListener ((UserEventListener)this, (ResourceClient)this, repository) ;
        em.addResourceStatusEventListener (this) ;
    }
    /**
     * methods defined by the UserEventListener interface.
     */
    public void UserEventReceived (UserEvent e) {
    }
    /**
     * Methods defined by the ResourceClient interface.
     */
    /**
     * In the case a cooperative application asks for an user event
     * exclusively used by me.
     */
    public boolean requestRelease(ResourceProxy proxy, Object requestData) {
        String name ;
        // let's retrieve the name of the repository, that I have created, and

```

```
// which contains the user event that the other application asks for.
name = (RepositoryDescriptor)proxy.getName () ;
if ((name.compareTo ("R1") == 0) & (myStatus == ...)) {
    // Ok I release this event.
    return true ;
} else {
    // No I need this event, sorry !
    return false ;
}
}
public void release (ResourceProxy proxy) {
    ...
}
public void notifyRelease (ResourceProxy proxy) {
    ...
}
public void statusChanged (ResourceStatusEvent event) {
    ...
}
}
```

# Package org.dvb.event

## Description

Provides access to user input events before they are processed through the event mechanism of the java.awt package.

Class Summary	
<b>Interfaces</b>	
<code>UserEventListener</code>	The listener interface for receiving user inputs.
<b>Classes</b>	
<code>EventManager</code>	The event manager allows an application to receive events coming from the user.
<code>OverallRepository</code>	This class defines a repository which contains all the user events defined in the <code>UserEvent</code> class.
<code>RepositoryDescriptor</code>	An instance of this class will be sent to clients of the DVB event API to notify them (through the interface <code>org.davic.resources.ResourceClient</code> ) when they are about to lose, or have lost, access to an event source.
<code>UserEvent</code>	Represents a user event.
<code>UserEventRepository</code>	The application will use this class to define the events that it wants to receive.

# org.dvb.event EventManager

## Syntax

```
public final class EventManager implements org.davic.resources.ResourceServer
java.lang.Object
|
+--org.dvb.event.EventManager
```

## All Implemented Interfaces:

org.davic.resources.ResourceServer

## Description

The event manager allows an application to receive events coming from the user. These events can be sent exclusively to an application or can be shared between applications. The Event Manager allows also the application to ask for exclusive access to some events, these events being received either from the standard java.awt event mechanism or by the mechanism defined in this package. The EventManager is a singleton.

The right to receive events is considered as the same resource regardless of whether it is being handled exclusively or shared. An application successfully obtaining exclusive access to an event results in all other applications losing access to that event, whether the access of those applications was shared or exclusive.

## Methods

---

### addExclusiveAccessToAWTEvent(ResourceClient, UserEventRepository)

```
public boolean addExclusiveAccessToAWTEvent(org.davic.resources.ResourceClient client,
      UserEventRepository userEvents)
```

An application should use this method to express its intend to have exclusive access to some events, but for these events to be received through the java.awt mechanism. The events the application wishes to receive are defined by the means of the UserEventRepository class. This repository is resolved at the time when this method call is made and adding or removing events from the repository after this method call doesn't affect the subscription to those events. An exclusive event will be sent to the application if this latest is focused.

#### Parameters:

`client` - resource client.

`userEvents` - the user events the application wants to be inform of.

#### Returns:

true if the events defined in the repository have been acquired, false otherwise.

#### Throws:

`IllegalArgumentException` - if the client argument is set to null.

---

### addResourceStatusEventListener(ResourceStatusListener)

```
public void addResourceStatusEventListener(org.davic.resources.ResourceStatusListener
      listener)
```

Adds the specified resource status listener so that an application can be aware of any changes regarding exclusive access to some events.

**Specified By:**

`org.davic.resources.ResourceServer.addResourceStatusEventListener(org.davic.resources.ResourceStatusListener)` in interface `org.davic.resources.ResourceServer`

**Parameters:**

`listener` - the resource status listener.

**addUserEventListener(UserEventListener, ResourceClient, UserEventRepository)**

```
public boolean addUserEventListener(UserEventListener listener,
    org.davic.resources.ResourceClient client, UserEventRepository userEvents)
```

Adds the specified listener to receive events coming from the user in an exclusive manner. The events the application wishes to receive are defined by the means of the `UserEventRepository` class. This repository is resolved at the time when this method call is made and adding or removing events from the repository after this method call doesn't affect the subscription to those events. The `ResourceClient` parameter indicates that the application wants to have an exclusive access to the user event defined in the repository.

**Parameters:**

`listener` - the listener to receive the user events.

`client` - resource client.

`userEvents` - a class which contains the user events it wants to be informed of.

**Returns:**

true if the events defined in the repository have been acquired, false otherwise.

**Throws:**

`IllegalArgumentException` - if the client argument is set to null.

**addUserEventListener(UserEventListener, UserEventRepository)**

```
public void addUserEventListener(UserEventListener listener,
    UserEventRepository userEvents)
```

Adds the specified listener to receive events coming from the user. The events the application wishes to receive are defined by the means of the `UserEventRepository` class. This repository is resolved at the time when this method call is made and adding or removing events from the repository after this method call doesn't affect the subscription to those events.

**Parameters:**

`listener` - the listener to receive the user events.

`userEvents` - a class which contains the user events it wants to be informed of.

**getInstance()**

```
public static EventManager getInstance()
```

This method returns the sole instance of the `EventManager` class. The `EventManager` class is a singleton.

**Returns:**

the instance of the `EventManager`.

---

**removeExclusiveAccessToAWTEvent(ResourceClient)**

```
public void removeExclusiveAccessToAWTEvent(org.davic.resources.ResourceClient client)
```

The application should use this method to release its exclusive access to user events defined by the means of the `addExclusiveAccessToAWTEvent` method.

**Parameters:**

`client` - the client that is no longer interested in events previously registered.

---

**removeResourceStatusEventListener(ResourceStatusListener)**

```
public void removeResourceStatusEventListener(org.davic.resources.ResourceStatusListener listener)
```

Removes the specified resource status listener.

**Specified By:**

`org.davic.resources.ResourceServer.removeResourceStatusEventListener(org.davic.resources.ResourceStatusListener)` in interface `org.davic.resources.ResourceServer`

**Parameters:**

`listener` - the listener to remove.

---

**removeUserEventListener(UserEventListener)**

```
public void removeUserEventListener(UserEventListener listener)
```

Removes the specified listener so that it will no longer receives user events. If it is appropriate (i.e. the application has asked for an exclusive access), the exclusive access is lost.

**Parameters:**

`listener` - the user event listener.

# org.dvb.event OverallRepository

## Syntax

```
public class OverallRepository extends UserEventRepository
```

```
java.lang.Object
|
+--UserEventRepository
|
+--org.dvb.event.OverallRepository
```

## Description

This class defines a repository which contains all the user events defined in the UserEvent class. For example, this pre-defined repository could be used by an application, which requires a pin code from the user, in order to prevent another applications from receiving events.

## See Also:

[UserEvent](#)

## Constructors

---

### OverallRepository()

```
public OverallRepository()
```

The constructor for the repository.



# org.dvb.event RepositoryDescriptor

## Syntax

```
public class RepositoryDescriptor implements org.davic.resources.ResourceProxy
```

```
java.lang.Object  
|  
+--org.dvb.event.RepositoryDescriptor
```

## All Implemented Interfaces:

org.davic.resources.ResourceProxy

## Description

An instance of this class will be sent to clients of the DVB event API to notify them (through the interface `org.davic.resources.ResourceClient`) when they are about to lose, or have lost, access to an event source. This object can be used by the application to get the name of the repository from which it will no longer be able to receive events.

## Methods

---

### getClient()

```
public org.davic.resources.ResourceClient getClient()
```

Return the object which asked to be notified about withdrawal of the event source. This is the object passed as the `ResourceClient` to whichever of the various 'add' methods on `ResourceManager` was used by the application to express interest in this repository.

#### Specified By:

`org.davic.resources.ResourceProxy.getClient()` in interface `org.davic.resources.ResourceProxy`

#### Returns:

the object which asked to be notified about withdrawal of the event source

---

### getName()

```
public java.lang.String getName()
```

Returns the name of the repository to which the lost user event belongs.

#### Returns:

String the name of the repository.

# org.dvb.event UserEvent

## Syntax

```
public class UserEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.dvb.event.UserEvent
```

## All Implemented Interfaces:

```
java.io.Serializable
```

## Description

Represents a user event. A user event is defined by a family, a type and either a code or a character. Unless stated otherwise, all constants used in this class are defined in `java.awt.event.KeyEvent` and its parent classes.

## Fields

---

### UEF\_KEY\_EVENT

```
public static final int UEF_KEY_EVENT
```

the family for events that are coming from the remote control or from the keyboard.

## Constructors

---

### UserEvent(Object, int, char)

```
public UserEvent(java.lang.Object source, int family, char keyChar)
```

Constructor for a new `UserEvent` object representing a key being typed. This is the combination of a key being pressed and then being released. The type of `UserEvents` created with this constructor shall be `KEY_TYPED`. Key combinations which do not result in characters, such as action keys like F1, shall not generate `KEY_TYPED` events.

#### Parameters:

`source` - the `EventManager` which is the source of the event

`family` - the event family.

`keyChar` - the character typed

#### Since:

MHP 1.0.1

---

**UserEvent(Object, int, int, int, int)**

```
public UserEvent(java.lang.Object source, int family, int type, int code, int modifiers)
```

Constructor for a new UserEvent object representing a key being pressed.

**Parameters:**

`source` - the `EventManager` which is the source of the event

`family` - the event family.

`type` - the event type. Either one of `KEY_PRESSED` or `KEY_RELEASED`.

`code` - the event code. One of the constants whose name begins in "VK\_" defined in `java.ui.event.KeyEvent` or `org.havi.ui.event`.

`modifiers` - the modifiers active when the key was pressed. These have the same semantics as modifiers in `java.awt.event.KeyEvent`.

## Methods

---

**getCode()**

```
public int getCode()
```

Returns the event code. For `KEY_TYPED` events, the code is `VK_UNDEFINED`.

**Returns:**

an int representing the event code.

---

**getFamily()**

```
public int getFamily()
```

Returns the event family. Could be `UEF_KEY_EVENT`.

**Returns:**

an int representing the event family.

---

**getKeyChar()**

```
public char getKeyChar()
```

Returns the character associated with the key in this event. If no valid Unicode character exists for this key event, `keyChar` is `CHAR_UNDEFINED`.

**Returns:**

a character

**Since:**

MHP 1.0.1

---

**getModifiers()**

```
public int getModifiers()
```

Returns the modifiers flag for this event.

**Returns:**

the modifiers flag for this event

**Since:**  
MHP 1.0.1

---

### **getType()**

```
public int getType()
```

Returns the event type. Could be KEY\_PRESSED, KEY\_RELEASED or KEY\_TYPED.

**Returns:**  
an int representing the event type.

---

### **isAltDown()**

```
public boolean isAltDown()
```

Returns whether or not the Alt modifier is down on this event.

**Returns:**  
whether the Alt modifier is down on this event

**Since:**  
MHP 1.0.1

---

### **isControlDown()**

```
public boolean isControlDown()
```

Returns whether or not the Control modifier is down on this event.

**Returns:**  
whether the Control modifier is down on this event

**Since:**  
MHP 1.0.1

---

### **isMetaDown()**

```
public boolean isMetaDown()
```

Returns whether or not the Meta modifier is down on this event.

**Returns:**  
whether the Meta modifier is down on this event

**Since:**  
MHP 1.0.1

---

### **isShiftDown()**

```
public boolean isShiftDown()
```

Returns whether or not the Shift modifier is down on this event.

**Returns:**  
whether the Shift modifier is down on this event

**Since:**  
MHP 1.0.1

# org.dvb.event UserEventListener

## Syntax

```
public interface UserEventListener extends java.util.EventListener
```

## All Superinterfaces:

java.util.EventListener

## Description

The listener interface for receiving user inputs.

## Methods

---

### userEventReceived(UserEvent)

```
public void userEventReceived(UserEvent e)
```

Called by the platform when a user input is received.

#### Parameters:

e - the user input event which was received

# org.dvb.event UserEventRepository

## Syntax

```
public class UserEventRepository  
  
java.lang.Object  
|  
+--org.dvb.event.UserEventRepository
```

## Direct Known Subclasses:

[OverallRepository](#)

## Description

The application will use this class to define the events that it wants to receive. Events that are able to be put in the repository are defined in the `UserEvent` class.

Where a repository includes a `KeyPressed` type event without the `KeyReleased` type event for the same key code or vice versa then exclusive reservations shall be made for both event types but only the one requested shall be received by the listener. Where a repository includes a `KEY_TYPED` event without the corresponding `KEY_PRESSED` and `KEY_RELEASED` events (excluding `KEY_PRESSED` or `KEY_RELEASED` events for modifiers), when an exclusive reservations is requested, it shall also be made for those corresponding `KEY_PRESSED` and `KEY_RELEASED` events but only the requested event shall be received by the listener.

Repositories do not keep a count of the number of times a particular user event is added or removed. Repeatedly adding an event to a repository has no effect. Removing an event removes it regardless of the number of times it has been added. For example, `org.dvb.event.UserEventRepository.addUserEvent(UserEvent event)` does nothing in case that the event is already in the repository. Events are considered to be already in the repository if an event with the same triplet of family, type and code is already in the repository.

Repositories are resolved when they are passed into the methods of `EventManager`. Adding or removing events from the repository after those method calls does not affect the subscription to those events.

## See Also:

[UserEvent](#)

## Constructors

---

### UserEventRepository(String)

```
public UserEventRepository(java.lang.String name)
```

The method to construct a new `UserEventRepository`.

#### Parameters:

`name` - the name of the repository.

## Methods

---

**addAllArrowKeys()**

```
public void addAllArrowKeys()
```

Adds the key codes for the arrow keys (VK\_LEFT, VK\_RIGHT, VK\_UP, VK\_DOWN). Any key codes already in the repository will not be added again.

---

**addAllColourKeys()**

```
public void addAllColourKeys()
```

Adds the key codes for the colour keys (VK\_COLORED\_KEY\_0, VK\_COLORED\_KEY\_1, VK\_COLORED\_KEY\_2, VK\_COLORED\_KEY\_3). Any key codes already in the repository will not be added again.

---

**addAllNumericKeys()**

```
public void addAllNumericKeys()
```

Adds the key codes for the numeric keys (VK\_0, VK\_1, VK\_2, VK\_3, VK\_4, VK\_5, VK\_6, VK\_7, VK\_8, VK\_9). Any key codes already in the repository will not be added again.

---

**addKey(int)**

```
public void addKey(int keycode)
```

A shortcut to create a new key event type entry in the repository. If a key is already in the repository, this method has no effect.

**Parameters:**

keycode - the key code.

---

**addUserEvent(UserEvent)**

```
public void addUserEvent(UserEvent event)
```

Adds the given user event to the repository.

**Parameters:**

event - the user event to be added in the repository.

---

**getUserEvent()**

```
public UserEvent[] getUserEvent()
```

Returns the list of the user events that are in the repository.

**Returns:**

an array which contains the user events that are in the repository.

---

**removeAllArrowKeys()**

```
public void removeAllArrowKeys()
```

Removes the key codes for the arrow keys (VK\_LEFT, VK\_RIGHT, VK\_UP, VK\_DOWN). Key codes from this set which are not present in the repository will be ignored.

---

**removeAllColourKeys()**

```
public void removeAllColourKeys()
```

Removes the key codes for the colour keys (VK\_COLORED\_KEY\_0, VK\_COLORED\_KEY\_1, VK\_COLORED\_KEY\_2, VK\_COLORED\_KEY\_3). Key codes from this set which are not present in the repository will be ignored.

---

**removeAllNumericKeys()**

```
public void removeAllNumericKeys()
```

Remove the key codes for the numeric keys (VK\_0, VK\_1, VK\_2, VK\_3, VK\_4, VK\_5, VK\_6, VK\_7, VK\_8, VK\_9). Key codes from this set which are not present in the repository will be ignored.

---

**removeKey(int)**

```
public void removeKey(int keycode)
```

The method to remove a key from the repository. Removing a key which is not in the repository has no effect.

**Parameters:**

`keycode` - the key code.

---

**removeUserEvent(UserEvent)**

```
public void removeUserEvent(UserEvent event)
```

Remove a user event from the repository. Removing a user event which is not in the repository shall have no effect.

**Parameters:**

`event` - the event to be removed from the repository.



---

## **Annex J (normative): DVB-J persistent storage API**

# Package org.dvb.io.persistent

## Description

Provides extensions to the java.io package for access to files held in persistent storage.

### Class Summary

#### Classes

<code>FileAccessPermissions</code>	This class encapsulates file access permissions, world, Organisation and owner.
<code>FileAttributes</code>	This class encapsulates the attributes of a file stored in persistent storage.

# org.dvb.io.persistent FileAccessPermissions

## Syntax

```
public class FileAccessPermissions

java.lang.Object
|
+--org.dvb.io.persistent.FileAccessPermissions
```

## Description

This class encapsulates file access permissions, world, Organisation and owner. World means all applications authorised to access persistent storage. Owner means the application which created the file. Organisation is defined as applications with the same organisation id as defined elsewhere in this specification.

## Constructors

---

### FileAccessPermissions(boolean, boolean, boolean, boolean, boolean, boolean)

```
public FileAccessPermissions(boolean readWorldAccessRight, boolean writeWorldAccessRight,
    boolean readOrganisationAccessRight, boolean writeOrganisationAccessRight,
    boolean readApplicationAccessRight, boolean writeApplicationAccessRight)
```

This constructor encodes all the file access permissions as a set of booleans.

#### Parameters:

`readWorldAccessRight` - read access for all applications  
`writeWorldAccessRight` - write access for all applications  
`readOrganisationAccessRight` - read access for organisation  
`writeOrganisationAccessRight` - write access for organisation  
`readApplicationAccessRight` - read access for the owner  
`writeApplicationAccessRight` - write access for the owner

## Methods

---

### hasReadApplicationAccessRight()

```
public boolean hasReadApplicationAccessRight()
```

Query whether this permission includes read access for the owning application

#### Returns:

true if the owning application can have read access, otherwise false.

---

### hasReadOrganisationAccessRight()

```
public boolean hasReadOrganisationAccessRight()
```

Query whether this permission includes read access for the organisation

**Returns:**

true if applications in this organisation can have read access, otherwise false.

---

**hasReadWorldAccessRight()**

```
public boolean hasReadWorldAccessRight()
```

Query whether this permission includes read access for the world.

**Returns:**

true if all applications can have read access, otherwise false.

---

**hasWriteApplicationAccessRight()**

```
public boolean hasWriteApplicationAccessRight()
```

Query whether this permission includes write access for the owning application

**Returns:**

true if the owning application can have write access, otherwise false.

---

**hasWriteOrganisationAccessRight()**

```
public boolean hasWriteOrganisationAccessRight()
```

Query whether this permission includes write access for the organisation

**Returns:**

true if applications in this organisation can have read access, otherwise false.

---

**hasWriteWorldAccessRight()**

```
public boolean hasWriteWorldAccessRight()
```

Query whether this permission includes write access for the world.

**Returns:**

true if all applications can have write access, otherwise false.

---

**setPermissions(boolean, boolean, boolean, boolean, boolean, boolean)**

```
public void setPermissions(boolean ReadWorldAccessRight, boolean WriteWorldAccessRight,  
    boolean ReadOrganisationAccessRight, boolean WriteOrganisationAccessRight,  
    boolean ReadApplicationAccessRight, boolean WriteApplicationAccessRight)
```

This method allows to modify the permissions on this instance of the FileAccessPermission class.

**Parameters:**

ReadWorldAccessRight - read access for all applications

WriteWorldAccessRight - write access for all applications

ReadOrganisationAccessRight - read access for organisation

WriteOrganisationAccessRight - write access for organisation

ReadApplicationAccessRight - read access for the owner

WriteApplicationAccessRight - write access for the owner

# org.dvb.io.persistent FileAttributes

## Syntax

```
public class FileAttributes
|
|  java.lang.Object
|  |
|  |--org.dvb.io.persistent.FileAttributes
```

## Description

This class encapsulates the attributes of a file stored in persistent storage. The default attributes for a file are low priority, owner read / write only permissions and null expiration date.

## Fields

---

### PRIORITY\_HIGH

```
public static final int PRIORITY_HIGH
Value for use as a file priority.
```

---

### PRIORITY\_LOW

```
public static final int PRIORITY_LOW
Value for use as a file priority.
```

---

### PRIORITY\_MEDIUM

```
public static final int PRIORITY_MEDIUM
Value for use as a file priority.
```

## Methods

---

### getExpirationDate()

```
public java.util.Date getExpirationDate()
```

Returns the expiration date. It will return the value used by the platform, which need not be the same as the value set.

**Returns:**

the expiration date

---

### getFileAttributes(File)

```
public static FileAttributes getFileAttributes(java.io.File f)
```

Get the attributes of a file.

**Parameters:**

`f` - the file to use

**Returns:**

a copy of the attributes of a file

**Throws:**

`SecurityException` - if the application is denied access to the file or to directories needed to reach the file by security policy

`IOException` - if access to the file fails due to an IO error or if the file reference is not to a valid location in persistent storage

---

**getPermissions()**

```
public FileAccessPermissions getPermissions()
```

Returns the file access permissions

**Returns:**

the file access permissions

---

**getPriority()**

```
public int getPriority()
```

Returns the priority to use in persistent storage

**Returns:**

the priority

---

**setExpirationDate(Date)**

```
public void setExpirationDate(java.util.Date d)
```

Sets the expiration date. This field is a hint to the platform to identify the date after which a file is no longer useful as perceived by the application. The platform may choose to use a different date than the one given as a parameter.

**Parameters:**

`d` - the expiration date

---

**setFileAttributes(FileAttributes, File)**

```
public static void setFileAttributes(FileAttributes p, java.io.File f)
```

Associate a set of file attributes with a file.

**Parameters:**

`p` - the file attributes to use

`f` - the file to use

**Throws:**

`SecurityException` - if the application is either denied access to the file or directories needed to reach the file by security policy or is not authorised to modify the attributes of the file.

`IOException` - if access to the file fails due to an IO error or if the file reference is not to a valid location in persistent storage

---

**setPermissions(FileAccessPermissions)**

```
public void setPermissions(FileAccessPermissions p)
```

Sets the file access permissions.

**Parameters:**

p - the file access permissions

---

**setPriority(int)**

```
public void setPriority(int priority)
```

Sets the priority to use in persistent storage

**Parameters:**

priority - the priority to set

---

## Annex K (informative): Extensions

Private protocols and possibly APIs are not precluded and are outside of the scope of the MHP specification.

The addition of public or protected methods or fields to classes and interfaces in the `org.dvb` namespace is **not** allowed.



---

## **Annex L (normative): User Settings and Preferences API**

# Package org.dvb.user

## Description

Provides access to settings and preferences configured by the end-user.

Class Summary	
<b>Interfaces</b>	
<code>UserPreferenceChangeListener</code>	An application wishing to be informed of any change to a user preference implements this interface.
<b>Classes</b>	
<code>Facility</code>	A facility maps a preference's name to a single value or to an array of values.
<code>GeneralPreference</code>	This class defines a set of general preferences.
<code>Preference</code>	This abstract class defines the Preference object.
<code>UserPreferenceChangeEvent</code>	This class defines the event sent to appropriate listeners when a user preference has been changed.
<code>UserPreferenceManager</code>	The UserPreferenceManager class gives access to the user preference settings.
<code>UserPreferencePermission</code>	This class is for user preference and setting permissions.
<b>Exceptions</b>	
<code>UnsupportedPreferenceException</code>	Thrown when a non-supported preference is used.

# org.dvb.user Facility

## Syntax

```
public class Facility
```

```
java.lang.Object
```

```
|
```

```
+--org.dvb.user.Facility
```

## Description

A facility maps a preference's name to a single value or to an array of values. A facility enables an application to define the list of values supported for a specified preference. For example, if an application is available in English or French then it can create a Facility ("User Language", {"English", "French"}). When the application will retrieve the "User Language" from the general preference it will specify the associated facility in order to get a Preference which will contain a set a values compatible with those supported by the application.

## Constructors

---

### Facility(String, String)

```
public Facility(java.lang.String preference, java.lang.String value)
```

Creates a Facility with a single value. This facility can be used by an application to retrieve a preference compatible with its capabilities.

**Parameters:**

`preference` - a String representing the name of the preference.

`value` - a String representing the value of the preference.

---

### Facility(String, String[])

```
public Facility(java.lang.String preference, java.lang.String[] values)
```

Creates a Facility with a set of values. This facility can be used by an application to retrieve a preference compatible with its capabilities.

**Parameters:**

`preference` - a String representing the name of the preference.

`values` - an array of String representing the set of values.

# org.dvb.user GeneralPreference

## Syntax

```
public final class GeneralPreference extends Preference
```

```
java.lang.Object
|
+--Preference
|
+--org.dvb.user.GeneralPreference
```

## Description

This class defines a set of general preferences. These preferences are read from the receiver and each application (downloaded or not) can access them through the `UserPreferenceManager.read` method. The standardized preferences are "User Language", "Parental Rating", "User Name", "User Address", "User @", "Country Code", "Default Font Size".

When constructed, objects of this class are empty and have no values defined. Values may be added using the `add` methods inherited from the `Preference` class or by calling `UserPreferenceManager.read`.

The encodings of these standardized preferences are as follows.

- User Language: 3 letter ISO 639 language codes;
- Parental rating: string using the same encoding as returned by `javax.tv.service.guide.ContentRatingAdvisory.getDisplayText`;
- User name: name of the user, first name(s) first and last name last;
- User Address: postal address of the user, may contain multiple lines separated by carriage return characters (as defined in table D-4).
- User @: e-mail address of the user in the SMTP form as defined in RFC821;
- Country Code: two letter ISO 3166-1 country code;
- Default font size: preferred font size for normal body text expressed in points, decimal integer value encoded as a string (26 is the default; differing size indicates a preference of different font size than usual)

## Constructors

---

### GeneralPreference(String)

```
public GeneralPreference(java.lang.String name)
```

Constructs a `GeneralPreference` object. A general preference maps a preference name to a list of strings.

#### Parameters:

`name` - the general preference name.

#### Throws:

`IllegalArgumentException` - if the preference's name is not supported.

# org.dvb.user Preference

## Syntax

```
public abstract class Preference  
  
java.lang.Object  
|  
+--org.dvb.user.Preference
```

## Direct Known Subclasses:

[GeneralPreference](#)

## Description

This abstract class defines the Preference object. A Preference maps a name to a list of favourite values. The first element in the list is the favourite value for this preference.

## Constructors

---

### Preference()

```
protected Preference()
```

This protected constructor is only present to enable sub-classes of this one to be defined by the platform. It is not intended to be used by inter-operable applications.

---

### Preference(String, String)

```
public Preference(java.lang.String name, java.lang.String value)
```

Creates a new preference with the specified name and the specified value. This single value will be the favourite one for this preference.

#### Parameters:

- `name` - a String object representing the name of the preference.
- `value` - a String object representing the value of the preference.

---

### Preference(String, String[])

```
public Preference(java.lang.String name, java.lang.String[] value)
```

Creates a new preference with the specified name and the specified value set.

#### Parameters:

- `name` - a String object representing the name of the preference.
- `value` - an array of String objects representing the set of values for this preference ordered from the most favourite to the least favourite.

## Methods

---

**add(int, String)**

```
public void add(int position, java.lang.String value)
```

Adds a new value for this preference. The value is inserted at the specified position. If the value is already in the list then it is moved to the position specified. If the position is greater than the length of the list, then the value is added to the end of this list. If the position is negative, then the value is added to the beginning of this list.

**Parameters:**

`position` - an int representing the position in the list.  
`value` - a String representing the new value to insert.

---

**add(String)**

```
public void add(java.lang.String value)
```

Adds a new value for this preference. The value is added to the end of the list. If the value is already in the list then it is moved to the end of the list.

**Parameters:**

`value` - a String object representing the new value.

---

**add(String[])**

```
public void add(java.lang.String[] values)
```

Adds several new values for this preferences. The values are added to the end of the list in the same order as they are found in the array passed to this method. Any values already in the list are moved to the position in the list which they would have if they were not already present.

**Parameters:**

`values` - an array of strings representing the values to add

**Since:**

MHP 1.0.1

---

**getFavourites()**

```
public java.lang.String[] getFavourites()
```

Returns the list of favourite values for this preference. Returns an empty array if no value sets are defined for this preference.

**Returns:**

an array of String representing the favourite values for this preference.

---

**getMostFavourite()**

```
public java.lang.String getMostFavourite()
```

Returns the most favourite value for this preference, that is, the first element of the list.

**Returns:**

a String representing the favourite values Returns null if no value is defined for this preference.

---

**getName()**

```
public java.lang.String getName()
```

Returns the name of the preference.

**Returns:**

a String object representing the name of the preference.

---

**getPosition(String)**

```
public int getPosition(java.lang.String value)
```

Returns the position in the list of the specified value.

**Parameters:**

`value` - a String representing the value to look for.

**Returns:**

an integer representing the position of the value in the list counting from zero. If the value is not found then it returns -1.

---

**hasValue()**

```
public boolean hasValue()
```

Tests if this preference has at least one value set.

**Returns:**

true if this preference has at least one value set, false otherwise.

---

**remove(String)**

```
public void remove(java.lang.String value)
```

Removes the specified value from the list of favourites. If the value is not in the list then the method call has no effect.

**Parameters:**

`value` - a String representing the value to remove.

---

**removeAll()**

```
public void removeAll()
```

Removes all the values of a preference

**Since:**

MHP 1.0.1

---

**setMostFavourite(String)**

```
public void setMostFavourite(java.lang.String value)
```

Sets the most favourite value for this preference. If the value is already in the list, then it is moved to the head. If the value is not already in the list then it is added at the head.

**Parameters:**

`value` - the most favourite value

---

**toString()**

```
public java.lang.String toString()
```

Convert name and favourites to a String.

**Overrides:**

`java.lang.Object.toString()` in class `java.lang.Object`

**Returns:**

the preference name and favourites

---

org.dvb.user

# UnsupportedPreferenceException

## Syntax

```
public class UnsupportedPreferenceException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--org.dvb.user.UnsupportedPreferenceException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Thrown when a non-supported preference is used.

## Constructors

---

### UnsupportedPreferenceException()

```
public UnsupportedPreferenceException()
```

Constructs a `UnsupportedPreferenceException` with no detail message.

---

### UnsupportedPreferenceException(String)

```
public UnsupportedPreferenceException(java.lang.String a)
```

Constructs a `UnsupportedPreferenceException` with a detail message.

#### Parameters:

a - the detail message



org.dvb.user

# UserPreferenceChangeEvent

## Syntax

```
public class UserPreferenceChangeEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.dvb.user.UserPreferenceChangeEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This class defines the event sent to appropriate listeners when a user preference has been changed.

## Constructors

---

### UserPreferenceChangeEvent(String)

```
public UserPreferenceChangeEvent(java.lang.String preferenceName)
```

Constructs a new event.

#### Parameters:

preferenceName - the name of the modified preference.

## Methods

---

### getName()

```
public java.lang.String getName()
```

Returns the name of the modified Preference

#### Returns:

the Preference name.

org.dvb.user

# UserPreferenceChangeListener

## Syntax

```
public interface UserPreferenceChangeListener
```

## Description

An application wishing to be informed of any change to a user preference implements this interface.

## Methods

---

### receiveUserPreferenceChangeEvent(UserPreferenceChangeEvent)

```
public void receiveUserPreferenceChangeEvent (UserPreferenceChangeEvent e)
```

This method is called when a user preference changes.

#### Parameters:

e - the event notifying this event.

# org.dvb.user UserPreferenceManager

## Syntax

```
public class UserPreferenceManager  
  
java.lang.Object  
|  
+--org.dvb.user.UserPreferenceManager
```

## Description

The `UserPreferenceManager` class gives access to the user preference settings. This class provides a set of methods that allow an application to read or save user settings. It also provides a mechanism to notify applications when a preference has been modified. The value of a user setting, retrieved with the `read` method, is a copy of the value that is stored in the receiver. The `write` method, if authorized, overwrites the stored value.

## Methods

---

### **addUserPreferenceChangeListener(UserPreferenceChangeListener)**

```
public void addUserPreferenceChangeListener(UserPreferenceChangeListener l)
```

Adds a listener for changes in user preferences.

**Parameters:**

l - the listener to add.

---

### **getInstance()**

```
public static UserPreferenceManager getInstance()
```

Return an instance of the `UserPreferenceManager` for this application. Repeated calls to this method by the same application shall return the same instance.

**Returns:**

an instance of `UserPreferenceManager`

---

### **read(Preference)**

```
public void read(Preference p)
```

Allows an application to read a specified user preference.

**Parameters:**

p - an object representing the preference to read.

**Throws:**

`SecurityException` - if the calling application is denied access to this preference

---

### **read(Preference, Facility)**

```
public void read(Preference p, Facility facility)
```

Allows an application to read a specified user preference taking into account the facility defined by the application. If the intersection between the two sets of values is empty then the preference will have no value. If there is a mis-match between the name of the preference used when constructing the facility and the name of the preference used in this method then the preference will have no value.

**Parameters:**

- `p` - an object representing the preference to read.
- `facility` - the preferred values the application for the preference

**Throws:**

- `SecurityException` - if the calling application is denied access to this preference

---

**removeUserPreferenceChangeListener(UserPreferenceChangeListener)**

```
public void removeUserPreferenceChangeListener(UserPreferenceChangeListener l)
```

Removes a listener for changes in user preferences.

**Parameters:**

- `l` - the listener to remove.

---

**write(Preference)**

```
public void write(Preference p)
```

Saves the specified user preference. If this method succeeds then it will change the value of this preference for all future MHP applications.

**Parameters:**

- `p` - the preference to save.

**Throws:**

- `UnsupportedPreferenceException` - if the preference provided is not a standardized preference as defined for use with `GeneralPreference`.

- `java.lang.SecurityException` - if the application does not have permission to call this method

- `IOException` - if saving the preference fails for other reasons

# org.dvb.user UserPreferencePermission

## Syntax

```
public class UserPreferencePermission extends java.security.BasicPermission
```

```
java.lang.Object
|
+--java.security.Permission
|
+--java.security.BasicPermission
|
+--org.dvb.user.UserPreferencePermission
```

## All Implemented Interfaces:

java.security.Guard, java.io.Serializable

## Description

This class is for user preference and setting permissions. A UserPreferencePermission contains a name, but no actions list.

The permission name can either be "read" or "write". The "read" permission allows an application to read the user preferences and settings (using `UserPreferenceManager.read`) for which read access is not always granted. Access to the following settings/preferences is always granted: "User Language", "Parental Rating", "Default Font Size" and "Country Code"

The "write" permission allows an application to modify user preferences and settings (using `UserPreferenceManager.write`).

## Constructors

---

### UserPreferencePermission(String)

```
public UserPreferencePermission(java.lang.String name)
```

Creates a new UserPreferencePermission with the specified name. The name is the symbolic name of the UserPreferencePermission.

#### Parameters:

`name` - the name of the UserPreferencePermission

---

### UserPreferencePermission(String, String)

```
public UserPreferencePermission(java.lang.String name, java.lang.String actions)
```

Creates a new UserPreferencePermission object with the specified name. The name is the symbolic name of the UserPreferencePermission, and the actions String is unused and should be null. This constructor exists for use by the Policy object to instantiate new Permission objects.

#### Parameters:

`name` - the name of the UserPreferencePermission

`actions` - should be null.

---

## **Annex M (normative): SI Access API**

# Package org.dvb.si

## Description

Provides access to DVB service information.

## General Design

Many of the methods in this package use a common design template. Asynchronous method calls to retrieve data from the network all return an instance of the `SIRequest` class. Applications may use this to inquire about or terminate the retrieval operation. When the retrieval operation completes, an instance of a subclass of `SIRetrievalEvent` will be sent to the `SIRetrievalListener` which the application passed in as a parameter to the original method call to retrieve data from the network. When constructing these events, the platform shall provide as the request parameter to the event constructor, the same instance of the `SIRequest` class as was returned by the original method call which started the retrieval operation. This `SIRequest` instance shall be returned by the `getSource` method on such events.

Class Summary	
<b>Interfaces</b>	
<code>DescriptorTag</code>	This interface defines constants corresponding to the most common descriptor tags.
<code>PMTElementaryStream</code>	This interface represents an elementary stream of a service.
<code>PMTService</code>	This interface represents a particular service carried by a transport stream.
<code>PMTStreamType</code>	This interface defines the constants corresponding to the different stream types
<code>SIBouquet</code>	This interface (together with the <code>SITransportStreamBAT</code> interface) represents a sub-table of the Bouquet Association Table (BAT) describing a particular bouquet.
<code>SIEvent</code>	This interface represents a particular event within a service.
<code>SIInformation</code>	This interface groups the common features of <code>SIBouquet</code> , <code>SINetwork</code> , <code>SITransportStream</code> , <code>SIService</code> , <code>PMTService</code> , <code>SIEvent</code> , <code>SITime</code> and <code>PMTElementaryStream</code> .
<code>SIIterator</code>	Objects implementing <code>SIIterator</code> interface allow to browse through a set of SI objects.
<code>SIMonitoringListener</code>	This interface shall be implemented by using application classes in order to listen to changes in monitored SI objects.
<code>SIMonitoringType</code>	This interface defines the constants corresponding to the SI information type values in <code>SIMonitoringEvent</code> .
<code>SINetwork</code>	This interface (together with the <code>SITransportStreamNIT</code> interface) represents a sub-table of the Network Information Table (NIT) describing a particular network.
<code>SIRetrievalListener</code>	This interface shall be implemented by application classes in order to receive events about completion of SI requests.
<code>SIRunningStatus</code>	This interface defines the constants corresponding to the running status values for services and events.
<code>SIService</code>	This interface represents a particular service carried by a transport stream.

<b>Class Summary</b>	
<code>SIServiceType</code>	This interface defines constants corresponding to the different service types.
<code>SITime</code>	This interface represents the Time and Date Table (TDT) and the (optional) Time Offset Table (TOT).
<code>SITransportStream</code>	This interface is the base interface for representing information about transport streams.
<code>SITransportStreamBAT</code>	This interface represents information about transport streams that has been retrieved from a BAT table.
<code>SITransportStreamDescription</code>	This interface represents the Transport Stream Description Table (TSDT).
<code>SITransportStreamNIT</code>	This interface represents information about transport streams that has been retrieved from a NIT table.
<code>TextualServiceIdentifierQuery</code>	An interface that can be implemented by objects representing DVB services.
<b>Classes</b>	
<code>Descriptor</code>	This class represents a descriptor within a sub-table.
<code>SIDatabase</code>	This class represents the root of the SI information hierarchy.
<code>SILackOfResourcesEvent</code>	This event is sent in response to a SI retrieval request when the resources needed for retrieving the data are not available, e.g.
<code>SIMonitoringEvent</code>	Objects of this class are sent to listener objects of the using application to notify that a change in the monitored information has happened.
<code>SINotInCacheEvent</code>	This event is sent in response to a SI retrieval request when the request was made with the FROM_CACHE_ONLY mode and the requested data is not present in the cache.
<code>SIOBJECTNotInTableEvent</code>	This event is sent in response to a SI retrieval request when the SI table where the information about the requested object should be located has been retrieved but the requested object is not present in it.
<code>SIRequest</code>	Object instances of this class represent SI retrieval requests made by the application.
<code>SIRequestCancelledEvent</code>	This event is sent in response to a SI retrieval request when the request is cancelled with the SIRequest.cancelRequest method call.
<code>SIRetrievalEvent</code>	This class is the base class for events about completion of a SI retrieval request.
<code>SISuccessfulRetrieveEvent</code>	This event is sent in response to a SI retrieval request when the retrieve request was successfully completed.
<code>SITableNotFoundEvent</code>	This event is sent in response to a SI retrieval request when the SI table that should contain the requested information could not be retrieved.
<code>SITableUpdatedEvent</code>	This event is sent in response to a SI descriptor retrieval request when the table carrying the information about the object has been updated and the set of descriptors consistent with the old object can not be retrieved.
<code>SIUtil</code>	This class contains SI related utility functions.
<b>Exceptions</b>	
<code>SIException</code>	This class is the root of the SI exceptions hierarchy.
<code>SIIllegalArgumentException</code>	This exception is thrown when one or more of the arguments passed to a method are invalid (e.g.



**Class Summary**`SIInvalidPeriodException`

This exception is thrown when a specified period is invalid (for example, start time is after the end time)

# org.dvb.si Descriptor

## Syntax

```
public class Descriptor
{
    java.lang.Object
    |
    +--org.dvb.si.Descriptor
}
```

## Description

This class represents a descriptor within a sub-table.

A descriptor consist of three fields: a tag, a contentLength and the content.

The tag uniquely identifies the descriptor type. The content length indicates the number of bytes in the content. The content consists of an array of bytes of length content length. The data represented by the content is descriptor type dependent.

## See Also:

[DescriptorTag](#)

## Methods

---

### getBytesAt(int)

```
public byte getBytesAt(int index)
```

Get a particular byte within the descriptor content

#### Parameters:

`index` - index to the descriptor content. Value 0 corresponds to the first byte after the length field.

#### Returns:

The required byte

#### Throws:

`IndexOutOfBoundsException` - if `index < 0` or `index >= ContentLength`

---

### getContent()

```
public byte[] getContent()
```

Get a copy of the content of this descriptor (everything after the length field).

#### Returns:

a copy of the content of the descriptor

---

### getContentLength()

```
public short getContentLength()
```

This method returns the length of the descriptor content as coded in the length field of this descriptor.

**Returns:**

The length of the descriptor content.

---

**getTag()**

```
public short getTag()
```

Get the descriptor tag

**Returns:**

The descriptor tag (the most common values are defined in the DescriptorTag interface)

**See Also:**

[DescriptorTag](#)

# org.dvb.si DescriptorTag

## Syntax

```
public interface DescriptorTag
```

## Description

This interface defines constants corresponding to the most common descriptor tags.

## See Also:

[Descriptor](#)

## Fields

---

### BOUQUET\_NAME

```
public static final short BOUQUET_NAME
```

Constant value for the descriptor tag as specified in EN 300 468

---

### CA\_IDENTIFIER

```
public static final short CA_IDENTIFIER
```

Constant value for the descriptor tag as specified in EN 300 468

---

### CABLE\_DELIVERY\_SYSTEM

```
public static final short CABLE_DELIVERY_SYSTEM
```

Constant value for the descriptor tag as specified in EN 300 468

---

### COMPONENT

```
public static final short COMPONENT
```

Constant value for the descriptor tag as specified in EN 300 468

---

### CONTENT

```
public static final short CONTENT
```

Constant value for the descriptor tag as specified in EN 300 468

---

### COUNTRY\_AVAILABILITY

```
public static final short COUNTRY_AVAILABILITY
```

Constant value for the descriptor tag as specified in EN 300 468

---

**DATA\_BROADCAST**

public static final short DATA\_BROADCAST

Constant value for the descriptor tag as specified in EN 300 468

---

**EXTENDED\_EVENT**

public static final short EXTENDED\_EVENT

Constant value for the descriptor tag as specified in EN 300 468

---

**FREQUENCY\_LIST**

public static final short FREQUENCY\_LIST

Constant value for the descriptor tag as specified in EN 300 468

---

**LINKAGE**

public static final short LINKAGE

Constant value for the descriptor tag as specified in EN 300 468

---

**LOCAL\_TIME\_OFFSET**

public static final short LOCAL\_TIME\_OFFSET

Constant value for the descriptor tag as specified in EN 300 468

---

**MOSAIC**

public static final short MOSAIC

Constant value for the descriptor tag as specified in EN 300 468

---

**MULTILINGUAL\_BOUQUET\_NAME**

public static final short MULTILINGUAL\_BOUQUET\_NAME

Constant value for the descriptor tag as specified in EN 300 468

---

**MULTILINGUAL\_COMPONENT**

public static final short MULTILINGUAL\_COMPONENT

Constant value for the descriptor tag as specified in EN 300 468

---

**MULTILINGUAL\_NETWORK\_NAME**

public static final short MULTILINGUAL\_NETWORK\_NAME

Constant value for the descriptor tag as specified in EN 300 468

---

**MULTILINGUAL\_SERVICE\_NAME**

public static final short MULTILINGUAL\_SERVICE\_NAME

Constant value for the descriptor tag as specified in EN 300 468

---

**NETWORK\_NAME**

public static final short NETWORK\_NAME

Constant value for the descriptor tag as specified in EN 300 468

---

**NVOD\_REFERENCE**

public static final short NVOD\_REFERENCE

Constant value for the descriptor tag as specified in EN 300 468

---

**PARENTAL\_RATING**

public static final short PARENTAL\_RATING

Constant value for the descriptor tag as specified in EN 300 468

---

**PARTIAL\_TRANSPORT\_STREAM**

public static final short PARTIAL\_TRANSPORT\_STREAM

Constant value for the descriptor tag as specified in EN 300 468

---

**PRIVATE\_DATA\_SPECIFIER**

public static final short PRIVATE\_DATA\_SPECIFIER

Constant value for the descriptor tag as specified in EN 300 468

---

**SATELLITE\_DELIVERY\_SYSTEM**

public static final short SATELLITE\_DELIVERY\_SYSTEM

Constant value for the descriptor tag as specified in EN 300 468

---

**SERVICE**

public static final short SERVICE

Constant value for the descriptor tag as specified in EN 300 468

---

**SERVICE\_LIST**

public static final short SERVICE\_LIST

Constant value for the descriptor tag as specified in EN 300 468

---

**SERVICE\_MOVE**

public static final short SERVICE\_MOVE

Constant value for the descriptor tag as specified in EN 300 468

---

**SHORT\_EVENT**

```
public static final short SHORT_EVENT
```

Constant value for the descriptor tag as specified in EN 300 468

---

**SHORT\_SMOOTHING\_BUFFER**

```
public static final short SHORT_SMOOTHING_BUFFER
```

Constant value for the descriptor tag as specified in EN 300 468

---

**STREAM\_IDENTIFIER**

```
public static final short STREAM_IDENTIFIER
```

Constant value for the descriptor tag as specified in EN 300 468

---

**STUFFING**

```
public static final short STUFFING
```

Constant value for the descriptor tag as specified in EN 300 468

---

**SUBTITLING**

```
public static final short SUBTITLING
```

Constant value for the descriptor tag as specified in EN 300 468

---

**TELEPHONE**

```
public static final short TELEPHONE
```

Constant value for the descriptor tag as specified in EN 300 468

---

**TELETEXT**

```
public static final short TELETEXT
```

Constant value for the descriptor tag as specified in EN 300 468

---

**TERRESTRIAL\_DELIVERY\_SYSTEM**

```
public static final short TERRESTRIAL_DELIVERY_SYSTEM
```

Constant value for the descriptor tag as specified in EN 300 468

---

**TIME\_SHIFTED\_EVENT**

```
public static final short TIME_SHIFTED_EVENT
```

Constant value for the descriptor tag as specified in EN 300 468

---

**TIME\_SHIFTED\_SERVICE**

```
public static final short TIME_SHIFTED_SERVICE
```

Constant value for the descriptor tag as specified in EN 300 468

---

# org.dvb.si PMTElementaryStream

## Syntax

```
public interface PMTElementaryStream extends SIInformation
```

## All Superinterfaces:

[SIInformation](#)

## Description

This interface represents an elementary stream of a service.

For each running service there is a PMT describing the elementary streams of the service. An object that implements this interface represents one such elementary stream. Each object that implements the PMTElementaryStream interface is identified by the combination of the identifiers original\_network\_id, transport\_stream\_id, service\_id, component\_tag (or elementary\_PID).

## See Also:

[PMTService](#), [PMTStreamType](#)

## Methods

---

### getComponentTag()

```
public int getComponentTag()
```

Get the component tag identifier.

#### Returns:

The component tag. If the elementary stream does not have an associated component tag, this method returns -2.

---

### getDvbLocator()

```
public org.davic.net.dvb.DvbLocator getDvbLocator()
```

Gets a DvbLocator that identifies this elementary stream

#### Returns:

The DvbLocator of this elementary stream

---

### getElementaryPID()

```
public short getElementaryPID()
```

Get the elementary PID.

#### Returns:

The PID the data of elementary stream is sent on in the transport stream.



---

**getOriginalNetworkID()**

```
public int getOriginalNetworkID()
```

Get the original network identification identifier.

**Returns:**

The original network identification.

---

**getServiceID()**

```
public int getServiceID()
```

Get the service identification identifier.

**Returns:**

The service identification.

---

**getStreamType()**

```
public byte getStreamType()
```

Get the stream type of this elementary stream.

**Returns:**

The stream type (some of the possible values are defined in the PMTStreamType interface).

**See Also:**

[PMTStreamType](#)

---

**getTransportStreamID()**

```
public int getTransportStreamID()
```

Get the transport stream identification identifier.

**Returns:**

The transport stream identification.

# org.dvb.si PMTService

## Syntax

```
public interface PMTService extends SIInformation
```

## All Superinterfaces:

`SIInformation`

## Description

This interface represents a particular service carried by a transport stream. The information is retrieved from the PMT table.

Each object that implements the PMTService interface is identified by the combination of the following identifiers: original\_network\_id, transport\_stream\_id, service\_id.

## Methods

---

### getDvbLocator()

```
public org.davic.net.dvb.DvbLocator getDvbLocator()
```

Gets a DvbLocator that identifies this service

**Returns:**

The DvbLocator of this service

---

### getOriginalNetworkID()

```
public int getOriginalNetworkID()
```

Get the original network identification.

**Returns:**

The original network identification identifier.

---

### getPcrPid()

```
public int getPcrPid()
```

Get the PCR pid.

**Returns:**

The PCR pid.

---

### getServiceID()

```
public int getServiceID()
```

Get the service identification.

**Returns:**

The service identification identifier.

---

**getTransportStreamID()**

```
public int getTransportStreamID()
```

Get the transport stream identification.

**Returns:**

The transport stream identification identifier.

---

**retrievePMTElementaryStreams(short, Object, SIReivalListener, short[])**

```
public SIRequest retrievePMTElementaryStreams(short retrieveMode, java.lang.Object appData,  
      SIReivalListener listener, short[] somePMTDescriptorTags)
```

Retrieve information associated with the elementary streams which compose this service from the Program Map Table (PMT).

The SIIterator that is returned with the event when the request completes successfully will contain one or more objects that implement the PMTElementaryStream interface. If no matching object was found, the appropriate one of the following events is sent: ObjectNotInCacheEvent, ObjectNotInTableEvent or TableNotFoundEvent.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM\_CACHE\_ONLY), from the cache if available and if not from the stream (FROM\_CACHE\_OR\_STREAM), or always from the stream (FROM\_STREAM\_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIReivalListener that will receive the event informing about the completion of the request.

`somePMTDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If `somePMTDescriptorTags` is null, the application is not interested in descriptors. All non applicable tag values are ignored.

**Returns:**

An SIRequest object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid

**See Also:**

`SIRequest`, `SIReivalListener`, `PMTElementaryStream`

# org.dvb.si PMTStreamType

## Syntax

```
public interface PMTStreamType
```

## Description

This interface defines the constants corresponding to the different stream types

## See Also:

```
PMTElementaryStream, getStreamType()
```

## Fields

---

### MPEG1\_AUDIO

```
public static final byte MPEG1_AUDIO
```

Constant value for the stream type as specified in ISO/IEC 13818-1

---

### MPEG1\_VIDEO

```
public static final byte MPEG1_VIDEO
```

Constant value for the stream type as specified in ISO/IEC 13818-1

---

### MPEG2\_AUDIO

```
public static final byte MPEG2_AUDIO
```

Constant value for the stream type as specified in ISO/IEC 13818-1

---

### MPEG2\_VIDEO

```
public static final byte MPEG2_VIDEO
```

Constant value for the stream type as specified in ISO/IEC 13818-1

# org.dvb.si

# SIBouquet

## Syntax

```
public interface SIBouquet extends SIInformation
```

## All Superinterfaces:

`SIInformation`

## Description

This interface (together with the `SITransportStreamBAT` interface) represents a sub-table of the Bouquet Association Table (BAT) describing a particular bouquet.

Each object that implements the `SIBouquet` interface is identified by the identifier `bouquet_id`.

## See Also:

`SITransportStreamBAT`

## Methods

---

### `getBouquetID()`

```
public int getBouquetID()
```

Get the identification.

#### Returns:

The bouquet identification of this bouquet.

---

### `getDescriptorTags()`

```
public short[] getDescriptorTags()
```

This method defines extra semantics for the `SIInformation.getDescriptorTags` method. If the BAT sub-table on which this `SIBouquet` object is based consists of multiple sections, then this method returns the descriptor tags in the order they appear when concatenating the descriptor loops of the different sections.

#### Overrides:

`getDescriptorTags()` in interface `SIInformation`

#### Returns:

The tags of the descriptors actually broadcast for the object (identified by their tags).

#### See Also:

`SIInformation`, `getDescriptorTags()`

---

### `getName()`

```
public java.lang.String getName()
```

This method returns the name of this bouquet. The name is extracted from the `bouquet_name_descriptor` or optionally from the `multilingual_bouquet_name_descriptor`. When this

information is not available "" is returned. All control characters as defined in ETR 211 are ignored. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation

**Returns:**

The bouquet name of this bouquet.

**getShortBouquetName()**

```
public java.lang.String getShortBouquetName()
```

This method returns the short name (ETR 211) of this bouquet without emphasis marks. The name is extracted from the bouquet\_name\_descriptor or optionally from the multilingual\_bouquet\_name\_descriptor. When this information is not available "" is returned. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

**Returns:**

The short bouquet name of this bouquet.

**getSIServiceLocators()**

```
public org.davic.net.dvb.DvbLocator[] getSIServiceLocators()
```

Get a list of DvbLocators identifying the services that belong to the bouquet.

**Returns:**

An array of DvbLocators identifying the services

**See Also:**

`org.davic.net.dvb.DvbLocator`, `SIService`

**retrieveDescriptors(short, Object, SIRetrievalListener)**

```
public SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener)
```

This method defines extra semantics for the SIInformation.retrieveDescriptors method (first prototype). If the BAT sub-table on which this SIBouquet object is based consists of multiple sections, then this method returns the requested descriptors in the order they appear when concatenating the descriptor loops of the different sections.

**Overrides:**

`retrieveDescriptors(short, Object, SIRetrievalListener)` in interface `SIInformation`

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid

**See Also:**

`SIInformation`, `retrieveDescriptors(short, Object, SIRetrievalListener)`

**retrieveDescriptors(short, Object, SIRetrievalListener, short[])**

```
public SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, short[] someDescriptorTags)
```

This method defines extra semantics for the `SIInformation.retrieveDescriptors` method (second prototype). If the BAT sub-table on which this `SI Bouquet` object is based consists of multiple sections, then this method returns the requested descriptors in the order they appear when concatenating the descriptor loops of the different sections.

**Overrides:**

```
retrieveDescriptors(short, Object, SIRetrievalListener, short[]) in
interface SIInformation
```

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of tags for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid

**See Also:**

`SIInformation`, `retrieveDescriptors(short, Object, SIRetrievalListener, short[])`

**retrieveSIBouquetTransportStreams(short, Object, SIRetrievalListener, short[])**

```
public SIRequest retrieveSIBouquetTransportStreams(short retrieveMode,
    java.lang.Object appData, SIRetrievalListener listener,
    short[] someDescriptorTags)
```

Retrieve information associated with transport streams belonging to the bouquet.

The `SIIterator` that is returned with the event when the request completes successfully will contain one or more objects that implement the `SITransportStreamBAT` interface.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid

**See Also:**

`SIRequest`, `SIRetrievalListener`, `SITransportStreamBAT`, `DescriptorTag`



# org.dvb.si SIDatabase

## Syntax

```
public class SIDatabase
    java.lang.Object
    |
    +--org.dvb.si.SIDatabase
```

## Description

This class represents the root of the SI information hierarchy. There is one SIDatabase per network interface. In a system with a single network interface there is only one SIDatabase object.

## Methods

---

### addBouquetMonitoringListener(SIMonitoringListener, int)

```
public void addBouquetMonitoringListener(SIMonitoringListener listener, int bouquetId)
```

Initiate monitoring of the bouquet information. When the bouquet information changes, an event will be delivered to the registered listener object.

How the monitoring is performed is implementation dependent and especially does not necessarily need to be continuous. The event will be delivered as soon as the implementation notices the change which might have some delay relative to when the change was actually made in the stream due to resources for the monitoring being scheduled between the monitoring activities of different tables. This specification does not set any minimum requirements for monitoring of the SI tables. This is to be done at a best effort basis by the implementation and is entirely implementation dependent. The only requirement is that when an implementation detects a change, e.g. because a resident Navigator or an MHP application has retrieved some SI information from the stream, then these listeners are notified of the change.

The monitoring stops silently and permanently when the network interface with which this SIDatabase object is associated starts tuning to another transport stream.

#### Parameters:

`listener` - listener object that will receive events when a change in the information is detected.

`bouquetId` - bouquet identifier of the bouquet whose information will be monitored.

#### Throws:

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range)

#### See Also:

`SIMonitoringListener`, `SIMonitoringEvent`

---

### addEventPresentFollowingMonitoringListener(SIMonitoringListener, int, int, int)

```
public void addEventPresentFollowingMonitoringListener(SIMonitoringListener listener,
    int originalNetworkId, int transportStreamId, int serviceId)
```

Initiate monitoring of information in the EIT related to present and following events. When the information related to those events changes, an event will be delivered to the registered listener object.

The scope of the monitoring is determined by the original network identifier, transport stream identifier and service identifier. The listener will be notified about the change of the information in any present and following event within that scope.

How the monitoring is performed is implementation dependent and especially does not necessarily need to be continuous. The event will be delivered as soon as the implementation notices the change which might have some delay relative to when the change was actually made in the stream due to resources for the monitoring being scheduled between the monitoring activities of different tables. This specification does not set any minimum requirements for monitoring of the SI tables. This is to be done at a best effort basis by the implementation and is entirely implementation dependent. The only requirement is that when an implementation detects a change, e.g. because a resident Navigator or an MHP application has retrieved some SI information from the stream, then these listeners are notified of the change.

The monitoring stops silently and permanently when the network interface with which this SIDatabase object is associated starts tuning to another transport stream.

**Parameters:**

`listener` - listener object that will receive events when a change in the information is detected.  
`originalNetworkId` - original network identifier specifying the scope of the monitoring.  
`transportStreamId` - transport stream identifier specifying the scope of the monitoring.  
`serviceId` - service identifier specifying the scope of the monitoring

**Throws:**

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range)

**See Also:**

`SIMonitoringListener`, `SIMonitoringEvent`

---

**addEventScheduleMonitoringListener(SIMonitoringListener, int, int, int, Date, Date)**

```
public void addEventScheduleMonitoringListener(SIMonitoringListener listener,
        int originalNetworkId, int transportStreamId, int serviceId,
        java.util.Date startTime, java.util.Date endTime)
```

Initiate monitoring of information in the EIT related to scheduled events. When the information related to those events changes, an event will be delivered to the registered listener object.

The scope of the monitoring is determined by the original network identifier, transport stream identifier, service identifier, start time and end time of the schedule period. The listener will be notified about the change of the information in any scheduled event within that scope.

How the monitoring is performed is implementation dependent and especially does not necessarily need to be continuous. The event will be delivered as soon as the implementation notices the change which might have some delay relative to when the change was actually made in the stream due to resources for the monitoring being scheduled between the monitoring activities of different tables. This specification does not set any minimum requirements for monitoring of the SI tables. This is to be done at a best effort basis by the implementation and is entirely implementation dependent. The only requirement is that when an implementation detects a change, e.g. because a resident Navigator or an MHP application has retrieved some SI information from the stream, then these listeners are notified of the change.

The monitoring stops silently and permanently when the network interface with which this SIDatabase object is associated starts tuning to another transport stream.

**Parameters:**

`listener` - listener object that will receive events when a change in the information is detected.  
`originalNetworkId` - original network identifier specifying the scope of the monitoring.  
`transportStreamId` - transport stream identifier specifying the scope of the monitoring.

`serviceId` - service identifier specifying the scope of the monitoring

`startTime` - start time of the schedule period

`endTime` - end time of the schedule period

**Throws:**

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range)

`SIInvalidPeriodException` - thrown if end time is before start time

**See Also:**

`SIMonitoringListener`, `SIMonitoringEvent`

**addNetworkMonitoringListener(SIMonitoringListener, int)**

```
public void addNetworkMonitoringListener(SIMonitoringListener listener, int networkId)
```

Initiate monitoring of the network information. When the network information changes, an event will be delivered to the registered listener object.

How the monitoring is performed is implementation dependent and especially does not necessarily need to be continuous. The event will be delivered as soon as the implementation notices the change which might have some delay relative to when the change was actually made in the stream due to resources for the monitoring being scheduled between the monitoring activities of different tables. This specification does not set any minimum requirements for monitoring of the SI tables. This is to be done at a best effort basis by the implementation and is entirely implementation dependent. The only requirement is that when an implementation detects a change, e.g. because a resident Navigator or an MHP application has retrieved some SI information from the stream, then these listeners are notified of the change.

The monitoring stops silently and permanently when the network interface with which this SIDatabase object is associated starts tuning to another transport stream.

**Parameters:**

`listener` - listener object that will receive events when a change in the information is detected.

`networkId` - network identifier of the network whose information will be monitored.

**Throws:**

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range)

**See Also:**

`SIMonitoringListener`, `SIMonitoringEvent`

**addPMTServiceMonitoringListener(SIMonitoringListener, int, int, int)**

```
public void addPMTServiceMonitoringListener(SIMonitoringListener listener,
    int originalNetworkId, int transportStreamId, int serviceId)
```

Initiate monitoring of information in the PMT related to a service. When the information related to a service changes, an event will be delivered to the registered listener object.

How the monitoring is performed is implementation dependent and especially does not necessarily need to be continuous. The event will be delivered as soon as the implementation notices the change which might have some delay relative to when the change was actually made in the stream due to resources for the monitoring being scheduled between the monitoring activities of different tables. This specification does not set any minimum requirements for monitoring of the SI tables. This is to be done at a best effort basis by the implementation and is entirely implementation dependent. The only requirement is that when an implementation detects a change, e.g. because a resident Navigator or an MHP application has retrieved some SI information from the stream, then these listeners are notified of the change.

The monitoring stops silently and permanently when the network interface with which this SIDatabase object is associated starts tuning to another transport stream.

**Parameters:**

`listener` - listener object that will receive events when a change in the information is detected.  
`originalNetworkId` - original network identifier of the service  
`transportStreamId` - transport stream identifier of the service  
`serviceId` - service identifier specifying the service whose information will be monitored

**Throws:**

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range)

**See Also:**

`SIMonitoringListener`, `SIMonitoringEvent`

### **addServiceMonitoringListener(SIMonitoringListener, int, int)**

```
public void addServiceMonitoringListener(SIMonitoringListener listener,
    int originalNetworkId, int transportStreamId)
```

Initiate monitoring of information in the SDT related to services. When the information related to services changes, an event will be delivered to the registered listener object.

The scope of the monitoring is determined by the original network identifier and transport stream identifier. The listener will be notified about the change of the information in any service within that scope.

How the monitoring is performed is implementation dependent and especially does not necessarily need to be continuous. The event will be delivered as soon as the implementation notices the change which might have some delay relative to when the change was actually made in the stream due to resources for the monitoring being scheduled between the monitoring activities of different tables. This specification does not set any minimum requirements for monitoring of the SI tables. This is to be done at a best effort basis by the implementation and is entirely implementation dependent. The only requirement is that when an implementation detects a change, e.g. because a resident Navigator or an MHP application has retrieved some SI information from the stream, then these listeners are notified of the change.

The monitoring stops silently and permanently when the network interface with which this SIDatabase object is associated starts tuning to another transport stream.

**Parameters:**

`listener` - listener object that will receive events when a change in the information is detected.  
`originalNetworkId` - original network identifier specifying the scope of the monitoring.  
`transportStreamId` - transport stream identifier specifying the scope of the monitoring.

**Throws:**

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range)

**See Also:**

`SIMonitoringListener`, `SIMonitoringEvent`

### **getSIDatabase()**

```
public static SIDatabase[] getSIDatabase()
```

Return an array of SIDatabase objects (one object per network interface). In a system with one network interface, the length of this array will be one. The network interface of each SIDatabase is used as data source for all new data accessed by this SIDatabase or SIInformation instances obtained from it.

This is the first method to be called to access the DVB-SI API. The returned SIDatabase objects provide the access point to the DVB-SI information.

**Returns:**

An array of SIDatabase objects, one per network interface.

**removeBouquetMonitoringListener(SIMonitoringListener, int)**

```
public void removeBouquetMonitoringListener(SIMonitoringListener listener, int bouquetId)
```

Removes the registration of an event listener for bouquet information monitoring. If this method is called with a listener that is registered but not with the same identifiers of the SI objects as given in the parameters, the method shall fail silently and the listeners stays registered with those identifiers that it has been added.

**Parameters:**

`listener` - listener object that has previously been registered

`bouquetId` - bouquet identifier of the bouquet whose information has been requested to be monitored

**Throws:**

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range)

**See Also:**

`SIMonitoringListener`, `SIMonitoringEvent`

**removeEventPresentFollowingMonitoringListener(SIMonitoringListener, int, int, int)**

```
public void removeEventPresentFollowingMonitoringListener(SIMonitoringListener listener,
    int originalNetworkId, int transportStreamId, int serviceId)
```

Removes the registration of an event listener for monitoring information related to present and following events. If this method is called with a listener that is registered but not with the same identifiers of the SI objects as given in the parameters, the method shall fail silently and the listeners stays registered with those identifiers that it has been added.

**Parameters:**

`listener` - listener object that has previously been registered

`originalNetworkId` - original network identifier specifying the scope of the monitoring.

`transportStreamId` - transport stream identifier specifying the scope of the monitoring.

`serviceId` - service identifier specifying the scope of the monitoring

**Throws:**

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range)

**See Also:**

`SIMonitoringListener`, `SIMonitoringEvent`

**removeEventScheduleMonitoringListener(SIMonitoringListener, int, int, int)**

```
public void removeEventScheduleMonitoringListener(SIMonitoringListener listener,
    int originalNetworkId, int transportStreamId, int serviceId)
```

Removes the registration of an event listener for monitoring information related to scheduled events for all periods. If this method is called with a listener that is registered but not with the same identifiers of the SI objects as given in the parameters, the method shall fail silently and the listeners stays registered with those identifiers that it has been added.

**Parameters:**

`listener` - listener object that has previously been registered

originalNetworkId - original network identifier specifying the scope of the monitoring.  
 transportStreamId - transport stream identifier specifying the scope of the monitoring.  
 serviceId - service identifier specifying the scope of the monitoring

**Throws:**

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range)

**See Also:**

`SIMonitoringListener`, `SIMonitoringEvent`

**removeNetworkMonitoringListener(SIMonitoringListener, int)**

```
public void removeNetworkMonitoringListener(SIMonitoringListener listener, int networkId)
```

Removes the registration of an event listener for network information monitoring. If this method is called with a listener that is registered but not with the same identifiers of the SI objects as given in the parameter, the method shall fail silently and the listeners stays registered with those identifiers that it has been added.

**Parameters:**

listener - listener object that has previously been registered

networkId - network identifier of the network which is no longer to be monitored by the listener

**Throws:**

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range)

**See Also:**

`SIMonitoringListener`, `SIMonitoringEvent`

**removePMTServiceMonitoringListener(SIMonitoringListener, int, int, int)**

```
public void removePMTServiceMonitoringListener(SIMonitoringListener listener,
        int originalNetworkId, int transportStreamId, int serviceId)
```

Removes the registration of an event listener for monitoring information in the PMT related to a service. If this method is called with a listener that is registered but not with the same identifiers of the SI objects as given in the parameters, the method shall fail silently and the listeners stays registered with those identifiers that it has been added.

**Parameters:**

listener - listener object that has previously been registered

originalNetworkId - original network identifier of the service

transportStreamId - transport stream identifier of the service

serviceId - service identifier specifying the service whose information has been requested to be monitored

**Throws:**

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range)

**See Also:**

`SIMonitoringListener`, `SIMonitoringEvent`

**removeServiceMonitoringListener(SIMonitoringListener, int, int)**

```
public void removeServiceMonitoringListener(SIMonitoringListener listener,
        int originalNetworkId, int transportStreamId)
```

Removes the registration of an event listener for monitoring information related to services. If this method is called with a listener that is registered but not with the same identifiers of the SI objects as

given in the parameters, the method shall fail silently and the listeners stays registered with those identifiers that it has been added.

**Parameters:**

`listener` - listener object that has previously been registered

`originalNetworkId` - original network identifier specifying the scope of the monitoring.

`transportStreamId` - transport stream identifier specifying the scope of the monitoring.

**Throws:**

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range)

**See Also:**

`SIMonitoringListener`, `SIMonitoringEvent`

**retrieveActualSINetwork(short, Object, SIRetrievalListener, short[])**

```
public SIRequest retrieveActualSINetwork(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, short[] someDescriptorTags)
```

Retrieve information associated with the actual network. The actual network is the network carrying the transport stream currently selected by the network interface connected to this SIDatabase.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SINetwork interface. If no matching object was found, the appropriate one of the following events is sent: `ObjectNotInCacheEvent`, `ObjectNotInTableEvent` or `TableNotFoundEvent`

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid

**See Also:**

`SIRequest`, `SIRetrievalListener`, `SINetwork`, `DescriptorTag`

**retrieveActualSIServices(short, Object, SIRetrievalListener, short[])**

```
public SIRequest retrieveActualSIServices(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, short[] someDescriptorTags)
```

Retrieve information associated with the actual services. The actual services are the services in the transport stream currently selected by the network interface connected to this SIDatabase.



The `SIIterator` that is returned with the event when the request completes successfully will contain one or more objects that implement the `SIService` interface. If no matching object was found, the appropriate one of the following events is sent: `ObjectNotInCacheEvent`, `ObjectNotInTableEvent` or `TableNotFoundEvent`.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains `-1` as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. `0...255`) are ignored, except for the special meaning of `-1` as the only element in the array.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid

**See Also:**

`SIRequest`, `SIRetrievalListener`, `SIService`, `DescriptorTag`

**retrieveActualSITransportStream(short, Object, SIRetrievalListener, short[])**

```
public SIRequest retrieveActualSITransportStream(short retrieveMode,
        java.lang.Object appData, SIRetrievalListener listener,
        short[] someDescriptorTags)
```

Retrieve information associated with the actual transport stream. The actual transport stream is the transport stream currently selected by the network interface connected to this `SIDatabase`.

The `SIIterator` that is returned with the event when the request completes successfully will contain an object that implements the `SITransportStreamNIT` interface. If no matching object was found, the appropriate one of the following events is sent: `ObjectNotInCacheEvent` `ObjectNotInTableEvent` or `TableNotFoundEvent`.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains `-1` as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. `0...255`) are ignored, except for the special meaning of `-1` as the only element in the array.



**Returns:**

An SIRequest object

**Throws:**

[SIIllegalArgumentException](#) - thrown if the retrieveMode is invalid

**See Also:**

[SIRequest](#), [SIRetrievalListener](#), [SITransportStream](#), [DescriptorTag](#)

**retrievePMTElementaryStreams(short, Object, SIRetrievalListener, DvbLocator, short[])**

```
public SIRequest retrievePMTElementaryStreams(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, org.davic.net.dvb.DvbLocator dvbLocator,
    short[] someDescriptorTags)
```

Retrieve PMT elementary stream information associated with components of a service. The required component(s) can be specified by its DVB locator.

The SIIterator that is returned with the event when the request completes successfully will contain one or more objects that implement the PMTElementaryStream interface. If no matching object was found, the appropriate one of the following events is sent: [ObjectNotInCacheEvent](#), [ObjectNotInTableEvent](#) or [TableNotFoundEvent](#).

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache ([FROM\\_CACHE\\_ONLY](#)), from the cache if available and if not from the stream ([FROM\\_CACHE\\_OR\\_STREAM](#)), or always from the stream ([FROM\\_STREAM\\_ONLY](#)).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - [SIRetrievalListener](#) that will receive the event informing about the completion of the request.

`dvbLocator` - DVB Locator identifying the component(s) of a service. The locator may be more specific than identifying one or more service components, but this method will only use the parts starting from the beginning up to the component tags.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

**Returns:**

An SIRequest object

**Throws:**

[SIIllegalArgumentException](#) - thrown if the retrieveMode is invalid or if the locator is invalid and does not identify one or more service components

**See Also:**

[SIRequest](#), [SIRetrievalListener](#), [SIService](#), [DescriptorTag](#)

**retrievePMTElementaryStreams(short, Object, SIRetrievalListener, int, int, short[])**

```
public SIRequest retrievePMTElementaryStreams(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, int serviceId, int componentTag,
    short[] someDescriptorTags)
```

Retrieve PMT elementary stream information associated with components of a service from the actual transport stream of this SIDatabase object. The elementary streams can be specified by their

identification. When -1 is specified for `componentTag` then elementary streams shall be retrieved regardless of their component tag.

The `SIIterator` that is returned with the event when the request completes successfully will contain one or more objects that implement the `PMTElementaryStream` interface. If no matching object was found, the appropriate one of the following events is sent: `ObjectNotInCacheEvent`, `ObjectNotInTableEvent` or `TableNotFoundEvent`.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

`serviceId` - Identification of the elementary streams to be retrieved: service identifier

`componentTag` - Identification of the elementary streams to be retrieved: component tag (-1 means return elementary streams regardless of their component tag)

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid or the numeric identifiers are out of range

**See Also:**

`SIRequest`, `SIRetrievalListener`, `SIService`, `DescriptorTag`

**retrievePMTService(short, Object, SIRetrievalListener, DvbLocator, short[])**

```
public SIRequest retrievePMTService(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, org.davic.net.dvb.DvbLocator dvbLocator,
    short[] someDescriptorTags)
```

Retrieve PMT information associated with a service. The required service can be specified by its DVB locator.

The `SIIterator` that is returned with the event when the request completes successfully will contain an object that implements the `PMTService` interface. If no matching object was found, the appropriate one of the following events is sent: `ObjectNotInCacheEvent` `ObjectNotInTableEvent` or `TableNotFoundEvent`.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

`dvbLocator` - DVB Locator identifying the service. The locator may be more specific than identifying a service, but this method will only use the parts starting from the beginning up to the service id.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid or the locator is invalid and does not identify a service

**See Also:**

`SIRequest`, `SIRetrievalListener`, `SIService`, `DescriptorTag`

**retrievePMTServices(short, Object, SIRetrievalListener, int, short[])**

```
public SIRequest retrievePMTServices(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, int serviceId, short[] someDescriptorTags)
```

Retrieve PMT information associated with services from the actual transport stream of this `SIDatabase` object. The required services can be specified by their identification. When -1 is specified as `serviceId` then services shall be retrieved regardless of their service id.

The `SIIterator` that is returned with the event when the request completes successfully will contain one or more objects that implement the `PMTService` interface. If no matching object was found, the appropriate one of the following events is sent: `ObjectNotInCacheEvent`, `ObjectNotInTableEvent` or `TableNotFoundEvent`.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

`serviceId` - Identification of the services to be retrieved: service identifier (-1 means return services regardless of their service id)

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid or the numeric identifiers are out of range

**See Also:**

`SIRequest`, `SIRetrievalListener`, `SIService`, `DescriptorTag`

**retrieveSIBouquets(short, Object, SIRetrievalListener, int, short[])**

```
public SIRequest retrieveSIBouquets(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, int bouquetId, short[] someDescriptorTags)
```

Retrieve information associated with bouquets. A bouquet can be specified by its identification. When `bouquetId` is set to -1, all bouquets signalled in the BAT of the currently received transport stream on that network interface are retrieved.

The `SIIterator` that is returned with the event when the request completes successfully will contain one or more objects that implement the `SIBouquet` interface. If no matching object was found, the appropriate one of the following events is sent: `ObjectNotInCacheEvent`, `ObjectNotInTableEvent` or `TableNotFoundEvent`.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

`bouquetId` - Identifier of the bouquet to be retrieved or -1 for all bouquets signalled on the currently received transport stream.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid or the numeric identifiers are out of range

**See Also:**

`SIRequest`, `SIRetrievalListener`, `SIBouquet`, `DescriptorTag`

**retrieveSINetworks(short, Object, SIRetrievalListener, int, short[])**

```
public SIRequest retrieveSINetworks(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, int networkId, short[] someDescriptorTags)
```

Retrieve information associated with networks. A network can be specified by its identification. When `networkId` is set to -1, all networks signalled in NIT Actual and Other of the currently received `TransportStream` on that network interface shall be retrieved.

The `SIIterator` that is returned with the event when the request completes successfully will contain one or more objects that implement the `SINetwork` interface. If no matching object was found, the

appropriate one of the following events is sent: `ObjectNotInCacheEvent`, `ObjectNotInTableEvent` or `TableNotFoundEvent`.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

`networkId` - Identification of the network to be retrieved or -1 for all networks currently signalled.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid or the numeric identifiers are out of range

**See Also:**

`SIRequest`, `SIRetrievalListener`, `SINetwork`, `DescriptorTag`

**retrieveSIService(short, Object, SIRetrievalListener, DvbLocator, short[])**

```
public SIRequest retrieveSIService(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, org.davic.net.dvb.DvbLocator dvbLocator,
    short[] someDescriptorTags)
```

Retrieve information associated with a service. The required service can be specified by its DVB locator.

The `SIIterator` that is returned with the event when the request completes successfully will contain an object that implements the `SIService` interface. If no matching object was found, the appropriate one of the following events is sent: `ObjectNotInCacheEvent` `ObjectNotInTableEvent` or `TableNotFoundEvent`"

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

`dvbLocator` - DVB locator identifying the service. The locator may be more specific than identifying a service, but this method will only use the parts starting from the beginning up to the service id.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid or the locator is invalid and does not identify a service

**See Also:**

`SIRequest`, `SIRetrievalListener`, `SIService`, `DescriptorTag`

**retrieveSIServices(short, Object, SIRetrievalListener, int, int, int, short[])**

```
public SIRequest retrieveSIServices(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, int originalNetworkId, int transportStreamId,
    int serviceId, short[] someDescriptorTags)
```

Retrieve information associated with services. The required services can be specified by their identification. When -1 is specified for `transportStreamId` then services shall be retrieved regardless of their transport stream id. When -1 is specified for `serviceId` then services shall be retrieved regardless of their service id.

The `SIIterator` that is returned with the event when the request completes successfully will contain one or more objects that implement the `SIService` interface. If no matching object was found, the appropriate one of the following events is sent: `ObjectNotInCacheEvent`, `ObjectNotInTableEvent` or `TableNotFoundEvent`.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

`originalNetworkId` - Identification of the services to be retrieved: original network identifier

`transportStreamId` - Identification of the services to be retrieved: transport stream identifier (-1 means return services regardless of their transport stream id)

`serviceId` - Identification of the services to be retrieved: service identifier (-1 means return services regardless of their service id)

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid or the numeric identifiers are out of range

**See Also:**

`SIRequest`, `SIRetrievalListener`, `SIService`, `DescriptorTag`

**retrieveSITimeFromTDT(short, Object, SIRetrievalListener)**

```
public SIRequest retrieveSITimeFromTDT(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener)
```

Retrieve information associated with time from the Time and Date Table (TDT) from the actual transport stream.

The `SIIterator` that is returned with the event when the request completes successfully will contain an object that implements the `SITime` interface. If no matching object was found, the appropriate one of the following events is sent: `ObjectNotInCacheEvent` `ObjectNotInTableEvent` or `TableNotFoundEvent`.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid

**See Also:**

`SIRequest`, `SIRetrievalListener`, `SITime`

**retrieveSITimeFromTOT(short, Object, SIRetrievalListener, short[])**

```
public SIRequest retrieveSITimeFromTOT(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, short[] someDescriptorTags)
```

Retrieve information associated with time from the Time Offset Table (TOT) from the actual transport stream. The time information will be accompanied with offset information

The `SIIterator` that is returned with the event when the request completes successfully will contain an object that implements the `SITime` interface. If no matching object was found, the appropriate one of the following events is sent: `ObjectNotInCacheEvent` `ObjectNotInTableEvent` or `TableNotFoundEvent`.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.



`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid

**See Also:**

`SIRequest`, `SIRetrievalListener`, `SITime`

**retrieveSITransportStreamDescription(short, Object, SIRetrievalListener, short[])**

```
public SIRequest retrieveSITransportStreamDescription(short retrieveMode,
    java.lang.Object appData, SIRetrievalListener listener,
    short[] someDescriptorTags)
```

Retrieve the `SITransportStreamDescription` object representing the information of the TSDT table in the actual transport stream of this `SIDatabase` object.

The `SIIterator` that is returned with the event when the request completes successfully will contain an object that implements the `SITransportStreamDescription` interface. If no matching object was found, the appropriate one of the following events is sent: `ObjectNotInCacheEvent`, `ObjectNotInTableEvent` or `TableNotFoundEvent`.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid

**See Also:**

`SIRequest`, `SIRetrievalListener`, `SITransportStreamDescription`, `DescriptorTag`



# org.dvb.si SIEvent

## Syntax

```
public interface SIEvent extends SIInformation
```

## All Superinterfaces:

[SIInformation](#)

## Description

This interface represents a particular event within a service.

Each object that implements the SIEvent interface is defined by the combination of the identifiers original\_network\_id, transport\_stream\_id, service\_id, event\_id.

## See Also:

[SIService](#)

## Methods

---

### getContentNibbles()

```
public byte[] getContentNibbles()
```

This method returns the content nibbles related to the event. This information is extracted from the content\_descriptor. If this descriptor is not present an empty array is returned (array with length 0). The return value is an array, each array element describes one content nibble. In each nibble the level 1 content nibbles occupy the four most significant bits of the returned bytes, level 2 content nibbles the four least significant bits.

#### Returns:

The content nibbles related to the event; level 1 content nibbles occupy the four most significant bits of the returned bytes, level 2 content nibbles the four least significant bits.

---

### getDuration()

```
public long getDuration()
```

Get the duration of this event.

#### Returns:

The duration in milliseconds.

---

### getDvbLocator()

```
public org.davic.net.dvb.DvbLocator getDvbLocator()
```

Gets a DvbLocator that identifies this event.

#### Returns:

The DvbLocator of this event

---

**getEventID()**

```
public int getEventID()
```

Get the event identification.

**Returns:**

The event identification.

---

**getFreeCAMode()**

```
public boolean getFreeCAMode()
```

Get the free\_CA\_mode value for this event, false indicates none of the component streams of this event are scrambled.

**Returns:**

The free\_CA\_mode value.

---

**getLevel1ContentNibbles()**

```
public byte[] getLevel1ContentNibbles()
```

This method returns the level 1 content nibbles of this event. This information is extracted from the content\_descriptor. If this descriptor is not present an empty array is returned (array with length 0). The return value is an array, each array element describes one content nibble. In each nibble the data occupies the four least significant bits of the returned bytes with the four most significant bits set to 0.

**Returns:**

All level 1 content nibbles related to the event.

---

**getName()**

```
public java.lang.String getName()
```

This method returns the name of this event. The name is extracted from a short\_event\_descriptor. When this information is not available "" is returned. All control characters as defined in ETR 211 are ignored. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

**Returns:**

The event name of this event.

---

**getOriginalNetworkID()**

```
public int getOriginalNetworkID()
```

Get the original network identification identifier.

**Returns:**

The original network identification.

---

**getRunningStatus()**

```
public byte getRunningStatus()
```

Get the running status of this event.

**Returns:**

The running status (the possible values are defined in the SIRunningStatus interface).

---

**See Also:**

`SIRunningStatus`

---

**getServiceID()**

```
public int getServiceID()
```

Get the service identification identifier.

**Returns:**

The service identification.

---

**getShortDescription()**

```
public java.lang.String getShortDescription()
```

This method returns the description of this event. The description is extracted from a `short_event_descriptor`. When this information is not available, "" is returned. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation

**Returns:**

The short description of this event.

---

**getShortEventName()**

```
public java.lang.String getShortEventName()
```

This method returns the short event name (ETR 211) of this event without emphasis marks. The name is extracted from a `short_event_descriptor`. When this information is not available "" is returned. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

**Returns:**

The short event name of this event.

---

**getStartTime()**

```
public java.util.Date getStartTime()
```

Get the start time of this event in UTC time.

**Returns:**

The start time of this event.

---

**getTransportStreamID()**

```
public int getTransportStreamID()
```

Get the transport stream identification identifier.

**Returns:**

The transport stream identification.

---

**retrieveSIService(short, Object, SIRetrievalListener, short[])**

```
public SIRequest retrieveSIService(short retrieveMode, java.lang.Object appData,  
    SIRetrievalListener listener, short[] someDescriptorTags)
```

This method retrieves the SIService object representing the service the event, represented by this SIEvent, is part of.

The `SIIterator` that is returned with the event when the request completes successfully will contain an object that implements the `SIService` interface. If no matching object was found, the appropriate one of the following events is sent: `ObjectNotInCacheEvent`, `ObjectNotInTableEvent` or `TableNotFoundEvent`.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains `-1` as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. `0...255`) are ignored, except for the special meaning of `-1` as the only element in the array.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid

**See Also:**

`SIRequest`, `SIRetrievalListener`, `SIService`, `DescriptorTag`

# org.dvb.si SIException

## Syntax

```
public abstract class SIException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--org.dvb.si.SIException
```

## Direct Known Subclasses:

```
SIIllegalArgumentException, SIInvalidPeriodException
```

## All Implemented Interfaces:

```
java.io.Serializable
```

## Description

This class is the root of the SI exceptions hierarchy.

## Constructors

---

### SIException()

```
public SIException()
```

Default constructor for the exception

---

### SIException(String)

```
public SIException(java.lang.String reason)
```

Constructor for the SI exception with a specified reason

#### Parameters:

`reason` - the reason why the exception was raised

# org.dvb.si SIIllegalArgumentException

## Syntax

```
public class SIIllegalArgumentException extends SIIException
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--SIIException
            |
            +--org.dvb.si.SIIllegalArgumentException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This exception is thrown when one or more of the arguments passed to a method are invalid (e.g. numeric identifiers out of range, etc.)

## Constructors

---

### SIIllegalArgumentException()

```
public SIIllegalArgumentException()
```

Default constructor for the exception

---

### SIIllegalArgumentException(String)

```
public SIIllegalArgumentException(java.lang.String reason)
```

Constructor for the exception with a specified reason

#### Parameters:

`reason` - the reason why the exception was raised

# org.dvb.si

# SIInformation

## Syntax

```
public interface SIInformation
```

## All Known Subinterfaces:

```
PMTElementaryStream, PMTService, SIBouquet, SIEvent, SINetwork, SIService,
SITime, SITransportStream, SITransportStreamBAT,
SITransportStreamDescription, SITransportStreamNIT
```

## Description

This interface groups the common features of SIBouquet, SINetwork, SITransportStream, SIService, PMTService, SIEvent, SITime and PMTElementaryStream.

Each SIInformation instance represents a sub-table (part). Any method accessing descriptors will retrieve descriptors from the same sub-table version as the SIInformation instance. When this version is no longer available, an SITableUpdatedEvent is returned.

## See Also:

```
SIBouquet, SINetwork, SITransportStream, SIService, PMTService, SIEvent, SITime,
PMTElementaryStream
```

## Fields

---

### FROM\_CACHE\_ONLY

```
public static final short FROM_CACHE_ONLY
```

Constant for retrieve mode parameter of the retrieve methods. When FROM\_CACHE\_ONLY mode is specified, the data will be retrieved only if it is in the cache. Otherwise, SINotInCacheEvent will be delivered to the listener. No stream access is done in this case.

---

### FROM\_CACHE\_OR\_STREAM

```
public static final short FROM_CACHE_OR_STREAM
```

Constant for retrieve mode parameter of the retrieve methods. When FROM\_CACHE\_OR\_STREAM mode is specified, the data will be retrieved from cache if it is present in the cache, otherwise it will be retrieved from the stream.

---

### FROM\_STREAM\_ONLY

```
public static final short FROM_STREAM_ONLY
```

Constant for retrieve mode parameter of the retrieve methods. When FROM\_STREAM\_ONLY mode is specified, the data will be retrieved directly from the stream and no cache access is tried first. This mode is meaningful only if the application knows that the information is not in the cache or that the information in the cache is no longer valid, but the implementation of the SI database may not be aware of the invalidity of the cached data. If the application has got the notification of the existence of

an updated version through the listener mechanism in this API, the implementation of the SI database is aware of the version change and the application should specify the FROM\_CACHE\_OR\_STREAM mode to be able to retrieve the data faster if the updated version has already been loaded to the cache by the SI database implementation.

## Methods

---

### fromActual()

```
public boolean fromActual()
```

Return true when the information contained in the object that implements this interface was filtered from an 'actual' table or from a table with no 'actual/other' distinction.

**Returns:**

true if the information comes from an 'actual' table or from a table with no 'actual/other' distinction, otherwise returns false

---

### getDataSource()

```
public org.davic.mpeg.TransportStream getDataSource()
```

Return the org.davic.mpeg.TransportStream object the information contained in the object that implements that interface was filtered from.

**Returns:**

The org.davic.mpeg.TransportStream object the information was filtered from.

**See Also:**

org.davic.mpeg.TransportStream

---

### getDescriptorTags()

```
public short[] getDescriptorTags()
```

Get the tags of all descriptors that are part of this version of this object. The tags are returned in the same order as the descriptors are broadcast. This method returns also the tags of descriptors that were not hinted at and that are not necessarily present in the cache. If there are no descriptors associated with this SIInformation object, this method returns an empty array whose length is 0.

**Returns:**

The tags of the descriptors actually broadcast for the object (identified by their tags).

**See Also:**

[DescriptorTag](#)

---

### getSIDatabase()

```
public SIDatabase getSIDatabase()
```

Return the root of the hierarchy the object that implements this interface belongs to.

**Returns:**

The root of the hierarchy.



---

**getUpdateTime()**

```
public java.util.Date getUpdateTime()
```

Return the time when the information contained in the object that implements this interface was last updated.

**Returns:**

The date of the last update.

---

**retrieveDescriptors(short, Object, SIRetrievalListener)**

```
public SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener)
```

This method retrieves all descriptors in the order the descriptors are broadcast.

This method is asynchronous and the completion of the method will be signalled by an `SISuccessfulRetrieveEvent` being sent to listener. Any retrieved descriptors are found in the `SIIterator` returned by the `getResult` method of that event. If descriptors are found then this iterator will contain `Descriptor` objects. If there are no matching descriptors, this iterator will contain no objects.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid

**See Also:**

`SIRequest`, `SIRetrievalListener`, `Descriptor`, `SIIterator`

---

**retrieveDescriptors(short, Object, SIRetrievalListener, short[])**

```
public SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, short[] someDescriptorTags)
```

Retrieve a set of descriptors. This method retrieves all or a set of descriptors in the order the descriptors are broadcast.

The tag values included in the `someDescriptorParameters` parameter are used for filtering the descriptors that are returned. Only those descriptors whose tag value is included in the `someDescriptorParameters` array are retrieved, unless the `someDescriptorParameters` array contains -1 as its one and only item in which case all descriptors related to this object are retrieved.

If the list of tags is a subset of the one hinted to the underlying implementation (in the request which created the object on which the method is called), this is likely to increase the efficiency of the (optional) caching mechanism

This method is asynchronous and the completion of the method will be signalled by an `SISuccessfulRetrieveEvent` being sent to listener. Any retrieved descriptors are found in the `SIIterator` returned by the `getResult` method of that event. If descriptors are found then this iterator will contain `Descriptor` objects. If there are no matching descriptors, this iterator will contain no objects.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

`someDescriptorTags` - - Descriptor tag values of descriptors that are used for filtering descriptors from the descriptors included in the SI table item corresponding to this `SIInformation` object. If the array contains -1 as its one and only element, all descriptors related to this object are retrieved.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid

**See Also:**

`SIRequest`, `SIRetrievalListener`, `Descriptor`, `SIIterator`, `DescriptorTag`

# org.dvb.si SIInvalidPeriodException

## Syntax

```
public class SIInvalidPeriodException extends SIException
```

```
java.lang.Object
|
+--java.lang.Throwable
   |
   +--java.lang.Exception
      |
      +--SIException
         |
         +--org.dvb.si.SIInvalidPeriodException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This exception is thrown when a specified period is invalid (for example, start time is after the end time)

## Constructors

---

### SIInvalidPeriodException()

```
public SIInvalidPeriodException()
```

Default constructor for the exception

---

### SIInvalidPeriodException(String)

```
public SIInvalidPeriodException(java.lang.String reason)
```

Constructor for the exception with a specified reason

#### Parameters:

`reason` - the reason why the exception was raised

# org.dvb.si SIIterator

## Syntax

```
public interface SIIterator extends java.util.Enumeration
```

## All Superinterfaces:

java.util.Enumeration

## Description

Objects implementing SIIterator interface allow to browse through a set of SI objects. In order to maintain consistency within the set of SI objects, this browsing does NOT initiate an actual access to the stream.

## Methods

---

### numberOfRemainingObjects()

```
public int numberOfRemainingObjects()
```

Get the number of remaining objects in the iterator.

#### Returns:

The number of remaining objects.

# org.dvb.si SILackOfResourcesEvent

## Syntax

```
public class SILackOfResourcesEvent extends SIRetrievalEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--SIRetrievalEvent
|
+--org.dvb.si.SILackOfResourcesEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This event is sent in response to a SI retrieval request when the resources needed for retrieving the data are not available, e.g. due to the necessary resources being all taken up by the calling application or other applications.

## See Also:

[SIRetrievalListener](#)

## Constructors

---

### SILackOfResourcesEvent(Object, SIRequest)

```
public SILackOfResourcesEvent(java.lang.Object appData, SIRequest request)
```

The constructor for the event

#### Parameters:

`appData` - the application data passed in the request method call

`request` - the SIRequest instance which is the source of the event

# org.dvb.si SIMonitoringEvent

## Syntax

```
public class SIMonitoringEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.dvb.si.SIMonitoringEvent
```

## All Implemented Interfaces:

```
java.io.Serializable
```

## Description

Objects of this class are sent to listener objects of the using application to notify that a change in the monitored information has happened.

## See Also:

[SIMonitoringType](#), [SIMonitoringListener](#)

## Constructors

---

### SIMonitoringEvent(SIDatabase, byte, int, int, int, int, int, Date, Date)

```
public SIMonitoringEvent(SIDatabase source, byte objectType, int networkId, int bouquetId,
    int originalNetworkId, int transportStreamId, int serviceId,
    java.util.Date startTime, java.util.Date endTime)
```

Constructor for the event object

#### Parameters:

- source - the SIDatabase object which is the source of the event
- objectType - type of the SIInformation object (constants in SIMonitoringType)
- networkId - networkId
- bouquetId - bouquetId
- originalNetworkId - originalNetworkId
- transportStreamId - transportStreamId
- serviceId - serviceId
- startTime - start time of event schedule period
- endTime - end time of event schedule period

## Methods

---

**getBouquetID()**

```
public int getBouquetID()
```

Returns the bouquetId of the bouquet. This method is only applicable if the SIInformation type returned with the getSIInformationType method is BOUQUET.

**Returns:**

the bouquetId or -2 if not applicable for this event

---

**getEndTime()**

```
public java.util.Date getEndTime()
```

Returns the end time of the schedule period whose event information has changed. This method is only applicable if the SIInformation type returned with the getSIInformationType method is SCHEDULED\_EVENT.

**Returns:**

the end time or null if not applicable for this event

---

**getNetworkID()**

```
public int getNetworkID()
```

Returns the networkId of the network. This method is only applicable if the SIInformation type returned with the getSIInformationType method is NETWORK.

**Returns:**

the networkId or -2 if not applicable for this event

---

**getOriginalNetworkID()**

```
public int getOriginalNetworkID()
```

Returns the originalNetworkId of the SIInformation objects This method is only applicable if the SIInformation type returned with the getSIInformationType method is SERVICE, PMT\_SERVICE, PRESENT\_FOLLOWING\_EVENT or SCHEDULED\_EVENT.

**Returns:**

the originalNetworkId or -2 if not applicable for this event

---

**getServiceID()**

```
public int getServiceID()
```

Returns the serviceId of the SIInformation objects This method is only applicable if the SIInformation type returned with the getSIInformationType method is PMT\_SERVICE, PRESENT\_FOLLOWING\_EVENT or SCHEDULED\_EVENT.

**Returns:**

the serviceId or -2 if not applicable for this event

---

**getSIInformationType()**

```
public byte getSIInformationType()
```

Get the SIInformation type of the information that has changed

**Returns:**

The SIInformation type (the possible values are defined in the SIMonitoringType interface).

---

**See Also:**

`SIMonitoringType`

---

**getSource()**

```
public java.lang.Object getSource()
```

Gets the SIDatabase instance that is sending the event.

**Overrides:**

`java.util.EventObject.getSource()` in class `java.util.EventObject`

**Returns:**

the SIDatabase instance that is the source of this event.

---

**getStartTime()**

```
public java.util.Date getStartTime()
```

Returns the start time of the schedule period whose event information has changed. This method is only applicable if the SIInformation type returned with the `getSIInformationType` method is `SCHEDULED_EVENT`.

**Returns:**

the start time or null if not applicable for this event

---

**getTransportStreamID()**

```
public int getTransportStreamID()
```

Returns the `transportStreamId` of the SIInformation objects. This method is only applicable if the SIInformation type returned with the `getSIInformationType` method is `SERVICE`, `PMT_SERVICE`, `PRESENT_FOLLOWING_EVENT` or `SCHEDULED_EVENT`.

**Returns:**

the `transportStreamId` or -2 if not applicable for this event

---



# org.dvb.si SIMonitoringListener

## Syntax

```
public interface SIMonitoringListener extends java.util.EventListener
```

## All Superinterfaces:

java.util.EventListener

## Description

This interface shall be implemented by using application classes in order to listen to changes in monitored SI objects.

## See Also:

[SIMonitoringEvent](#)

## Methods

---

### postMonitoringEvent(SIMonitoringEvent)

```
public void postMonitoringEvent(SIMonitoringEvent anEvent)
```

This method is called back by the SI API implementation to notify the listener about an event.

#### Parameters:

`anEvent` - The notified event.

#### See Also:

[SIMonitoringEvent](#)

# org.dvb.si

# SIMonitoringType

## Syntax

```
public interface SIMonitoringType
```

## Description

This interface defines the constants corresponding to the SI information type values in SIMonitoringEvent.

## See Also:

[SIMonitoringListener](#), [SIMonitoringEvent](#)

## Fields

---

### BOUQUET

```
public static final byte BOUQUET
```

Constant for the type of SIInformation object: Bouquet

---

### NETWORK

```
public static final byte NETWORK
```

Constant for the type of SIInformation object: Network

---

### PMT\_SERVICE

```
public static final byte PMT_SERVICE
```

Constant for the type of SIInformation object: PMTService

---

### PRESENT\_FOLLOWING\_EVENT

```
public static final byte PRESENT_FOLLOWING_EVENT
```

Constant for the type of SIInformation object: Present or following event

---

### SCHEDULED\_EVENT

```
public static final byte SCHEDULED_EVENT
```

Constant for the type of SIInformation object: Scheduled event

---

### SERVICE

```
public static final byte SERVICE
```

Constant for the type of SIInformation object: Service

# org.dvb.si

# SINetwork

## Syntax

```
public interface SINetwork extends SIInformation
```

## All Superinterfaces:

```
SIInformation
```

## Description

This interface (together with the SITransportStreamNIT interface) represents a sub-table of the Network Information Table (NIT) describing a particular network.

Each object that implements the SINetwork interface is identified by the identifier `network_id`.

## See Also:

```
SITransportStream, SITransportStreamNIT
```

## Methods

---

### getDescriptorTags()

```
public short[] getDescriptorTags()
```

This method defines extra semantics for the SIInformation.getDescriptorTags method. If the NIT sub-table on which this SINetwork object is based consists of multiple sections, then this method returns the descriptor tags in the order they appear when concatenating the descriptor loops of the different sections.

#### Overrides:

```
getDescriptorTags() in interface SIInformation
```

#### Returns:

The tags of the descriptors actually broadcast for the object (identified by their tags).

#### See Also:

```
SIInformation, getDescriptorTags()
```

---

### getName()

```
public java.lang.String getName()
```

This method returns the name of this network. The name is extracted from the `network_name_descriptor` or optionally from the `multilingual_network_name_descriptor`. When this information is not available "" is returned. All control characters as defined in ETR 211 are ignored. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

#### Returns:

The network name of this network.

---

**getNetworkID()**

```
public int getNetworkID()
```

Get the identification of this network.

**Returns:**

The network identification identifier.

---

**getShortNetworkName()**

```
public java.lang.String getShortNetworkName()
```

This method returns the short name (ETR 211) of this network without emphasis marks. The name is extracted from the `network_name_descriptor` or optionally from the `multilingual_network_name_descriptor`. When this information is not available "" is returned. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

**Returns:**

The short network name of this network.

---

**retrieveDescriptors(short, Object, SIReivalListener)**

```
public SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData,
    SIReivalListener listener)
```

This method defines extra semantics for the `SIInformation.retrieveDescriptors` method (first prototype). If the NIT sub-table on which this `SINetwork` object is based consists of multiple sections, then this method returns the requested descriptors in the order they appear when concatenating the descriptor loops of the different sections.

**Overrides:**

```
retrieveDescriptors(short, Object, SIReivalListener) in interface
SIInformation
```

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIReivalListener` that will receive the event informing about the completion of the request.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid

**See Also:**

`SIInformation`, `retrieveDescriptors(short, Object, SIReivalListener)`

---

**retrieveDescriptors(short, Object, SIReivalListener, short[])**

```
public SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData,
    SIReivalListener listener, short[] someDescriptorTags)
```

This method defines extra semantics for the `SIInformation.retrieveDescriptors` method (second prototype). If the NIT sub-table on which this `SINetwork` object is based consists of multiple sections, then this method returns the requested descriptors in the order they appear when concatenating the descriptor loops of the different sections.

**Overrides:**

```
retrieveDescriptors(short, Object, SIRetrievalListener, short[]) in
interface SIInformation
```

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of tags for descriptors (identified by their tags) the application is interested in. If the array contains `-1` as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. `0...255`) are ignored, except for the special meaning of `-1` as the only element in the array.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid

**See Also:**

```
SIInformation, retrieveDescriptors(short, Object, SIRetrievalListener,
short[])
```

**retrieveSITransportStreams(short, Object, SIRetrievalListener, short[])**

```
public SIRequest retrieveSITransportStreams(short retrieveMode, java.lang.Object appData,
SIRetrievalListener listener, short[] someDescriptorTags)
```

Retrieve information associated with transport streams carried via the network.

The `SIIterator` that is returned with the event when the request completes successfully will contain one or more objects that implement the `SITransportStreamNIT` interface.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains `-1` as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All

values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

**Returns:**

An SIRequest object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid

**See Also:**

`SIRequest`, `SIRetrievalListener`, `SITransportStreamNIT`, `DescriptorTag`

# org.dvb.si SINotInCacheEvent

## Syntax

```
public class SINotInCacheEvent extends SIRetrievalEvent
```

```
java.lang.Object
|
+--java.util.EventObject
    |
    +--SIRetrievalEvent
        |
        +--org.dvb.si.SINotInCacheEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This event is sent in response to a SI retrieval request when the request was made with the FROM\_CACHE\_ONLY mode and the requested data is not present in the cache.

## See Also:

[SIRetrievalListener](#)

## Constructors

---

### SINotInCacheEvent(Object, SIRequest)

```
public SINotInCacheEvent(java.lang.Object appData, SIRequest request)
```

The constructor for the event

#### Parameters:

`appData` - the application data passed in the request method call

`request` - the SIRequest instance which is the source of the event

# org.dvb.si SIOBJECTNOTINTABLEEVENT

## Syntax

```
public class SIOBJECTNOTINTABLEEVENT extends SIRETRIEVALEVENT
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--SIRETRIEVALEVENT
|
+--org.dvb.si.SIOBJECTNOTINTABLEEVENT
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This event is sent in response to a SI retrieval request when the SI table where the information about the requested object should be located has been retrieved but the requested object is not present in it. The reason may be that the object corresponding to the requested identifiers does not exist.

## See Also:

[SIRETRIEVALLISTENER](#)

## Constructors

---

### SIOBJECTNOTINTABLEEVENT(Object, SIREQUEST)

```
public SIOBJECTNOTINTABLEEVENT(java.lang.Object appData, SIREQUEST request)
```

The constructor for the event

#### Parameters:

`appData` - the application data passed in the request method call

`request` - the SIREQUEST instance which is the source of the event



# org.dvb.si SIRequest

## Syntax

```
public class SIRequest
    java.lang.Object
    |
    +--org.dvb.si.SIRequest
```

## Description

Object instances of this class represent SI retrieval requests made by the application. The application may cancel the request using this object.

## Methods

---

### cancelRequest()

```
public boolean cancelRequest()
```

Cancels the retrieval request.

**Returns:**

true if the request was cancelled and an `SIRequestCancelledEvent` will be delivered to the listener, false if the request has already completed (either successfully, with an error or due to a prior cancel method call)

---

### isAvailableInCache()

```
public boolean isAvailableInCache()
```

Returns whether the information will be returned from cache or from the stream

**Returns:**

true if the information will be returned from cache, false if the information will be retrieved from the stream

# org.dvb.si SIRequestCancelledEvent

## Syntax

```
public class SIRequestCancelledEvent extends SIRetrievalEvent
```

```
java.lang.Object
|
+--java.util.EventObject
    |
    +--SIRetrievalEvent
        |
        +--org.dvb.si.SIRequestCancelledEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This event is sent in response to a SI retrieval request when the request is cancelled with the `SIRequest.cancelRequest` method call.

## See Also:

`SIRequest`, `SIRetrievalListener`

## Constructors

---

### SIRequestCancelledEvent(Object, SIRequest)

```
public SIRequestCancelledEvent(java.lang.Object appData, SIRequest request)
```

The constructor for the event

#### Parameters:

`appData` - the application data passed in the request method call

`request` - the `SIRequest` instance which is the source of the event

# org.dvb.si

# SIRetrievalEvent

## Syntax

```
public abstract class SIRetrievalEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.dvb.si.SIRetrievalEvent
```

## Direct Known Subclasses:

```
SILackOfResourcesEvent, SINotInCacheEvent, SIOBJECTNotInTableEvent,
SIRequestCancelledEvent, SISuccessfulRetrieveEvent, SITableNotFoundEvent,
SITableUpdatedEvent
```

## All Implemented Interfaces:

```
java.io.Serializable
```

## Description

This class is the base class for events about completion of a SI retrieval request. Exactly one event will be returned in response to an SI retrieval request.

## See Also:

```
SIRetrievalListener
```

## Constructors

---

### SIRetrievalEvent(Object, SIRequest)

```
public SIRetrievalEvent(java.lang.Object appData, SIRequest request)
```

The constructor for the event

#### Parameters:

`appData` - the application data passed in the request method call

`request` - the SIRequest instance which is the source of the event

## Methods

---

### getAppData()

```
public java.lang.Object getAppData()
```

Returns the application data that was passed to the retrieve method

#### Returns:

the application data

---

**getSource()**

```
public java.lang.Object getSource()
```

Returns the SIRequest object that is the source of this event

**Overrides:**

java.util.EventObject.getSource() in class java.util.EventObject

**Returns:**

the SIRequest object

# org.dvb.si SIRetrievalListener

## Syntax

```
public interface SIRetrievalListener extends java.util.EventListener
```

## All Superinterfaces:

java.util.EventListener

## Description

This interface shall be implemented by application classes in order to receive events about completion of SI requests.

## See Also:

[SIRetrievalEvent](#)

## Methods

---

### postRetrievalEvent(SIRetrievalEvent)

```
public void postRetrievalEvent(SIRetrievalEvent event)
```

This method is called by the SI API implementation to notify the listener about completion of an SI request.

#### Parameters:

`event` - The event object.

#### See Also:

[SIRetrievalEvent](#)

# org.dvb.si

# SIRunningStatus

## Syntax

```
public interface SIRunningStatus
```

## Description

This interface defines the constants corresponding to the running status values for services and events.

## Fields

---

### NOT\_RUNNING

```
public static final byte NOT_RUNNING
```

Constant value for the running status as specified in EN 300 468

---

### PAUSING

```
public static final byte PAUSING
```

Constant value for the running status as specified in EN 300 468

---

### RUNNING

```
public static final byte RUNNING
```

Constant value for the running status as specified in EN 300 468

---

### STARTS\_IN\_A\_FEW\_SECONDS

```
public static final byte STARTS_IN_A_FEW_SECONDS
```

Constant value for the running status as specified in EN 300 468

---

### UNDEFINED

```
public static final byte UNDEFINED
```

Constant value for the running status as specified in EN 300 468

# org.dvb.si

# SIService

## Syntax

```
public interface SIService extends SIInformation, TextualServiceIdentifierQuery
```

## All Superinterfaces:

```
SIInformation, TextualServiceIdentifierQuery
```

## Description

This interface represents a particular service carried by a transport stream. Information that can be obtained through the methods of this interface is retrieved from the SDT table.

Each object that implements the SIService interface is identified by the combination of the following identifiers: original\_network\_id, transport\_stream\_id, service\_id.

## Methods

---

### getDvbLocator()

```
public org.davic.net.dvb.DvbLocator getDvbLocator()
```

Gets a DvbLocator that identifies this service.

**Returns:**

The DvbLocator of this service

---

### getEITPresentFollowingFlag()

```
public boolean getEITPresentFollowingFlag()
```

Get the EIT\_present\_following\_flag value, true indicates this service has present and/or following event information.

**Returns:**

The EIT\_present\_following\_flag value.

---

### getEITScheduleFlag()

```
public boolean getEITScheduleFlag()
```

Get the EIT\_schedule\_flag value, true indicates this services has scheduled event information.

**Returns:**

The EIT\_schedule\_flag value.

---

### getFreeCAMode()

```
public boolean getFreeCAMode()
```

Retrieve the free\_CA\_mode value of this service, false indicates none of the components of this service are scrambled.

**Returns:**

The free\_CA\_mode value of this service.

---

**getName()**

```
public java.lang.String getName()
```

This method returns the name of the service represented by this service. The name is extracted from the service\_descriptor or optionally from the multilingual\_service\_name\_descriptor. If this descriptor is not present "" is returned. All control characters as defined in ETR 211 are ignored. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

**Returns:**

The name of this service.

---

**getOriginalNetworkID()**

```
public int getOriginalNetworkID()
```

Get the original network identification.

**Returns:**

The original network identification identifier.

---

**getProviderName()**

```
public java.lang.String getProviderName()
```

This method returns the service provider name of this service. The service provider name is extracted from the service\_descriptor or optionally from the multilingual\_service\_name\_descriptor. If this descriptor is not present "" is returned. All control characters as defined in ETR 211 are ignored. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

**Returns:**

The service provider name of this service.

---

**getRunningStatus()**

```
public byte getRunningStatus()
```

Retrieve the running status of this service.

**Returns:**

The running status (the possible values are defined in the SIRunningStatus interface)

**See Also:**

[SIRunningStatus](#)

---

**getServiceID()**

```
public int getServiceID()
```

Get the service identification.

**Returns:**

The service identification identifier.

---

**getShortProviderName()**

```
public java.lang.String getShortProviderName()
```



This method returns the short name (ETR 211) of the service provider of this service without emphasis marks. The name is extracted from the `service_descriptor` or optionally from the `multilingual_service_name_descriptor`. When this information is not available "" is returned. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

**Returns:**

The short service provider name of this service.

**getShortServiceName()**

```
public java.lang.String getShortServiceName()
```

This method returns the short name (ETR 211) of this service without emphasis marks. The name is extracted from the `service_descriptor` or optionally from the `multilingual_service_name_descriptor`. When this information is not available "" is returned. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

**Returns:**

The short name of this service.

**getSIServiceType()**

```
public short getSIServiceType()
```

Get the service type. The service type is extracted from the `service_descriptor`.

**Returns:**

The service type. (Some of the possible values are defined in the `SIServiceType` interface.)

**See Also:**

[SIServiceType](#)

**getTextualServiceIdentifiers()**

```
public java.lang.String[] getTextualServiceIdentifiers()
```

Returns the textual service identifiers related to this object.

**Overrides:**

`getTextualServiceIdentifiers()` in interface `TextualServiceIdentifierQuery`

**Returns:**

an array of String objects containing the textual service identifiers or null if none are present.

**Since:**

MHP1.0.1

**getTransportStreamID()**

```
public int getTransportStreamID()
```

Get the transport stream identification.

**Returns:**

The transport stream identification identifier.

**retrieveFollowingSIEvent(short, Object, SIRetrievalListener, short[])**

```
public SIRequest retrieveFollowingSIEvent(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, short[] someDescriptorTags)
```

Retrieve information associated with the following event from the EIT-present/following.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SIEvent interface. If no matching object was found, the appropriate one of the following events is sent: ObjectNotInCacheEvent ObjectNotInTableEvent or TableNotFoundEvent.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM\_CACHE\_ONLY), from the cache if available and if not from the stream (FROM\_CACHE\_OR\_STREAM), or always from the stream (FROM\_STREAM\_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptorTags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

**Returns:**

An SIRequest object

**Throws:**

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid

**See Also:**

`SIRequest`, `SIRetrievalListener`, `SIEvent`, `DescriptorTag`

**retrievePMTService(short, Object, SIRetrievalListener, short[])**

```
public SIRequest retrievePMTService(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, short[] someDescriptorTags)
```

Retrieve the PMTService information associated with this service.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the PMTService interface. If no matching object was found, the appropriate one of the following events is sent: ObjectNotInCacheEvent ObjectNotInTableEvent or TableNotFoundEvent.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM\_CACHE\_ONLY), from the cache if available and if not from the stream (FROM\_CACHE\_OR\_STREAM), or always from the stream (FROM\_STREAM\_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptorTags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

**Returns:**

An SIRequest object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid

**See Also:**

`SIRequest`, `SIRetrievalListener`, `PMTService`, `DescriptorTag`

**retrievePresentSIEvent(short, Object, SIRetrievalListener, short[])**

```
public SIRequest retrievePresentSIEvent(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, short[] someDescriptorTags)
```

Retrieve information associated with the present event from the EIT-present/following.

The `SIIterator` that is returned with the event when the request completes successfully will contain an object that implements the `SIEvent` interface. If no matching object was found, the appropriate one of the following events is sent: `ObjectNotInCacheEvent`, `ObjectNotInTableEvent` or `TableNotFoundEvent`.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid

**See Also:**

`SIRequest`, `SIRetrievalListener`, `SIEvent`, `DescriptorTag`

**retrieveScheduledSIEvents(short, Object, SIRetrievalListener, short[], Date, Date)**

```
public SIRequest retrieveScheduledSIEvents(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, short[] someDescriptorTags,
    java.util.Date startTime, java.util.Date endTime)
```

Retrieve information associated with the scheduled events within the service for a requested period from the EIT-schedule. The events are presented in the order they are present in the EIT-schedule.

The `SIIterator` that is returned with the event when the request completes successfully will contain one or more objects that implement the `SIEvent` interface.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (`FROM_CACHE_ONLY`), from the cache if available and if not from the stream (`FROM_CACHE_OR_STREAM`), or always from the stream (`FROM_STREAM_ONLY`).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - `SIRetrievalListener` that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If `someDescriptorTags` is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

`startTime` - The beginning of the required period in UTC time.

`endTime` - The end of the required period in UTC time.

**Returns:**

An `SIRequest` object

**Throws:**

`SIIllegalArgumentException` - thrown if the `retrieveMode` is invalid

`SIInvalidPeriodException` - When no valid period is indicated.

**See Also:**

`SIRequest`, `SIRetrievalListener`, `SIEvent`, `DescriptorTag`

# org.dvb.si

# SIServiceType

## Syntax

```
public interface SIServiceType
```

## Description

This interface defines constants corresponding to the different service types.

## See Also:

```
getSIServiceType()
```

## Fields

---

### D\_D2\_MAC

```
public static final short D_D2_MAC
```

Constant value for the service type as specified in EN 300 468

---

### DATA\_BROADCAST

```
public static final short DATA_BROADCAST
```

Constant value for the service type as specified in EN 300 468

---

### DIGITAL\_RADIO\_SOUND

```
public static final short DIGITAL_RADIO_SOUND
```

Constant value for the service type as specified in EN 300 468

---

### DIGITAL\_TELEVISION

```
public static final short DIGITAL_TELEVISION
```

Constant value for the service type as specified in EN 300 468

---

### FM\_RADIO

```
public static final short FM_RADIO
```

Constant value for the service type as specified in EN 300 468

---

### MHP\_APPLICATION

```
public static final short MHP_APPLICATION
```

Constant value for the service type as specified in EN 300 468

---

**MOSAIC**

public static final short MOSAIC

Constant value for the service type as specified in EN 300 468

---

**NTSC**

public static final short NTSC

Constant value for the service type as specified in EN 300 468

---

**NVOD\_REFERENCE**

public static final short NVOD\_REFERENCE

Constant value for the service type as specified in EN 300 468

---

**NVOD\_TIME\_SHIFTED**

public static final short NVOD\_TIME\_SHIFTED

Constant value for the service type as specified in EN 300 468

---

**PAL**

public static final short PAL

Constant value for the service type as specified in EN 300 468

---

**SECAM**

public static final short SECAM

Constant value for the service type as specified in EN 300 468

---

**TELETEXT**

public static final short TELETEXT

Constant value for the service type as specified in EN 300 468

---

**UNKNOWN**

public static final short UNKNOWN

Constant value for the service type as specified in EN 300 468

---

# org.dvb.si SISuccessfulRetrieveEvent

## Syntax

```
public class SISuccessfulRetrieveEvent extends SIRetrievealEvent
```

```
java.lang.Object
|
+--java.util.EventObject
    |
    +--SIRetrievealEvent
        |
        +--org.dvb.si.SISuccessfulRetrieveEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This event is sent in response to a SI retrieval request when the retrieve request was successfully completed. The result of the request can be obtained from the getResult method.

## See Also:

[SIRetrievealListener](#)

## Constructors

---

### SISuccessfulRetrieveEvent(Object, SIRequest, SIIterator)

```
public SISuccessfulRetrieveEvent(java.lang.Object appData, SIRequest request,
                                SIIterator result)
```

The constructor for the event

#### Parameters:

- appData - the application data passed in the request method call
- request - the SIRequest instance which is the source of the event
- result - an SIIterator containing the retrieved objects

## Methods

---

### getResult()

```
public SIIterator getResult()
```

Returns the requested data in an SIIterator object.

#### Returns:

An SIIterator containing the requested objects

#### See Also:

[SIObjectNotInTableEvent](#)

# org.dvb.si SITableNotFoundEvent

## Syntax

```
public class SITableNotFoundEvent extends SIRetrievalEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--SIRetrievalEvent
|
+--org.dvb.si.SITableNotFoundEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This event is sent in response to a SI retrieval request when the SI table that should contain the requested information could not be retrieved. The reason may be that the requested table is not broadcast in the transport stream currently associated with the SI database.

## See Also:

[SIRetrievalListener](#)

## Constructors

---

### SITableNotFoundEvent(Object, SIRequest)

```
public SITableNotFoundEvent(java.lang.Object appData, SIRequest request)
```

The constructor for the event

#### Parameters:

`appData` - the application data passed in the request method call

`request` - the SIRequest instance which is the source of the event



# org.dvb.si SITableUpdatedEvent

## Syntax

```
public class SITableUpdatedEvent extends SIRetrievalEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--SIRetrievalEvent
|
+--org.dvb.si.SITableUpdatedEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This event is sent in response to a SI descriptor retrieval request when the table carrying the information about the object has been updated and the set of descriptors consistent with the old object can not be retrieved. The application should in this case first update the SIInformation object and then request the descriptors again.

## See Also:

[SIRetrievalListener](#)

## Constructors

---

### SITableUpdatedEvent(Object, SIRequest)

```
public SITableUpdatedEvent(java.lang.Object appData, SIRequest request)
```

The constructor for the event

#### Parameters:

`appData` - the application data passed in the request method call

`request` - the SIRequest instance which is the source of the event

# org.dvb.si

# SITime

## Syntax

```
public interface SITime extends SIInformation
```

## All Superinterfaces:

[SIInformation](#)

## Description

This interface represents the Time and Date Table (TDT) and the (optional) Time Offset Table (TOT). When it represents a TDT table, the `retrieveDescriptors` and `getDescriptorTags` methods behave as documented in the case when there are no descriptors, because the TDT does not contain any descriptors.

## See Also:

[SIDatabase](#)

## Methods

---

### **getUTCtime()**

```
public java.util.Date getUTCtime()
```

Get the UTC time as coded in the TDT or TOT table.

#### **Returns:**

The UTC as coded in the TDT or TOT table.

# org.dvb.si

# SITransportStream

## Syntax

```
public interface SITransportStream extends SIInformation
```

## All Known Subinterfaces:

```
SITransportStreamBAT, SITransportStreamNIT
```

## All Superinterfaces:

```
SIInformation
```

## Description

This interface is the base interface for representing information about transport streams.

Transport stream retrieval methods in the SIDatabase class and the SINetwork interface use the NIT table and will return objects that implement the SITransportStreamNIT interface.

Transport stream retrieval methods in the SIBouquet interface use the BAT table and will return objects that implement the SITransportStreamBAT interface.

## Methods

---

### getDvbLocator()

```
public org.davic.net.dvb.DvbLocator getDvbLocator()
```

Gets a DvbLocator that identifies this transport stream.

**Returns:**

The DvbLocator of this transport stream.

---

### getOriginalNetworkID()

```
public int getOriginalNetworkID()
```

Get the original network identification.

**Returns:**

The original network identification identifier.

---

### getTransportStreamID()

```
public int getTransportStreamID()
```

Get the transport stream identification.

**Returns:**

The transport stream identification identifier.

---

**retrieveSIServices(short, Object, SIRetrievalListener, short[])**

```
public SIRequest retrieveSIServices(short retrieveMode, java.lang.Object appData,  
    SIRetrievalListener listener, short[] someDescriptorTags)
```

Retrieve information associated with services carried via the transport stream. This method works in the same way for objects that implement the SITransportStreamNIT and SITransportStreamBAT interfaces.

The SIIterator that is returned with the event when the request completes successfully will contain objects that implement the SIService interface.

**Parameters:**

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM\_CACHE\_ONLY), from the cache if available and if not from the stream (FROM\_CACHE\_OR\_STREAM), or always from the stream (FROM\_STREAM\_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptorTags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

**Returns:**

An SIRequest object

**Throws:**

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid

**See Also:**

`SIRequest`, `SIRetrievalListener`, `SIService`, `DescriptorTag`

org.dvb.si

# SITransportStreamBAT

## Syntax

```
public interface SITransportStreamBAT extends SITransportStream
```

## All Superinterfaces:

`SIInformation`, `SITransportStream`

## Description

This interface represents information about transport streams that has been retrieved from a BAT table. All descriptor accessing methods return descriptors retrieved from a BAT table. Methods in `SIBouquet` for retrieving transport streams return objects that implement this interface.

## Methods

---

### **getBouquetID()**

```
public int getBouquetID()
```

Get the identification of the bouquet this transport stream is part of.

#### **Returns:**

The bouquet identification identifier.

org.dvb.si

# SITransportStreamDescription

## Syntax

```
public interface SITransportStreamDescription extends SIInformation
```

## All Superinterfaces:

[SIInformation](#)

## Description

This interface represents the Transport Stream Description Table (TSDT).

It defines no methods of its own other than those inherited from [SIInformation](#).

## See Also:

[SIDatabase](#), [SITransportStream](#)

org.dvb.si

# SITransportStreamNIT

## Syntax

```
public interface SITransportStreamNIT extends SITransportStream
```

## All Superinterfaces:

`SIInformation`, `SITransportStream`

## Description

This interface represents information about transport streams that has been retrieved from a NIT table. All descriptor accessing methods return descriptors retrieved from a NIT table. Methods in `SIDatabase` and `SINetwork` for retrieving transport streams return objects that implement this interface.

## Methods

---

### **getNetworkID()**

```
public int getNetworkID()
```

Get the identification of the network this transport stream is part of.

#### **Returns:**

The network identification identifier.

# org.dvb.si SIUtil

## Syntax

```
public class SIUtil  
  
java.lang.Object  
|  
+--org.dvb.si.SIUtil
```

## Description

This class contains SI related utility functions.

## Methods

---

### convertSIStringToJavaString(byte[], int, int, boolean)

```
public static java.lang.String convertSIStringToJavaString(byte[] dvbSIText, int offset,  
int length, boolean emphasizedPartOnly)
```

This method converts a text string that is coded according to annex A of the DVB-SI specification (EN 300 468) to a Java String object.

The text that must be converted is contained in 'dvbSIText' from index 'offset' to index 'offset+length-1' (inclusive).

If the text that must be converted is not validly coded according to annex A of the DVB-SI specification, then the result is undefined.

#### Parameters:

`dvbSIText` - The byte array that contains the string that must be converted.

`offset` - The offset indicates the start of the DVB-SI text in `dvbSIText`.

`length` - Length of the DVB-SI text in bytes.

`emphasizedPartOnly` - If `emphasizedPartOnly` is true, then only the text that is marked as emphasized (using the character emphasis on [0x86] and character emphasis off [0x87] control codes) will be returned. Otherwise, the character emphasis codes will be ignored, and all of the converted text will be returned.

#### Returns:

The converted text.

#### Throws:

`SIIllegalArgumentException` - thrown if `offset` and/or `offset+length-1` is not a valid index in `dvbSIText`.



org.dvb.si

# TextualServiceIdentifierQuery

## Syntax

```
public interface TextualServiceIdentifierQuery
```

## All Known Subinterfaces:

`SIService`

## Description

An interface that can be implemented by objects representing DVB services. Allows applications to obtain the textual service identifiers related to a service.

## Since:

MHP1.0.1

## Methods

---

### `getTextualServiceIdentifiers()`

```
public java.lang.String[] getTextualServiceIdentifiers()
```

Returns the textual service identifiers related to this object.

#### Returns:

an array of String objects containing the textual service identifiers or null if none are present.

#### Since:

MHP1.0.1

---

## **Annex N (normative): Streamed Media API Extensions**

# Package org.dvb.media

## Description

Provides DVB specific extensions to the Java Media Framework.

Class Summary	
<b>Interfaces</b>	
<code>BackgroundVideoPresentationControl</code>	A control to support the setting and querying of the video presentation for background players.
<code>SubtitleListener</code>	Report that a subtitle event has happened.
<code>SubtitlingEventControl</code>	Allow applications to register and unregister their interest in events related to the availability and presentation of subtitles.
<code>VideoFormatControl</code>	This provides a means for applications to get information associated with the format and aspect ratio of the video being presented to the user.
<code>VideoFormatListener</code>	The listener used to receive video format events
<code>VideoPresentationControl</code>	A control to support setting and querying the video presentation.
<b>Classes</b>	
<code>ActiveFormatDescriptionChangedEvent</code>	Event signalling that the transmitted active format definition has changed
<code>AspectRatioChangedEvent</code>	Event signalling that the aspect ratio of the transmitted video has changed
<code>CASStopEvent</code>	This event is generated whenever access to a service is withdrawn by the CA system, e.g.
<code>DFCChangedEvent</code>	Event signalling that the decoder format conversion being used has changed
<code>DripFeedDataSource</code>	This class allows to create a source for a JMF player to be able to feed the decoder progressively with parts of a clip (e.g.
<code>DripFeedPermission</code>	This class represents a permission to access the drip feed mode.
<code>NoComponentSelectedEvent</code>	This event is generated whenever presentation of a stream stops because there are no selected components to present.
<code>PresentationChangedEvent</code>	This event is generated whenever the content being presented by a player changes for reasons outside the control of the application.
<code>ServiceRemovedEvent</code>	This event is generated whenever access to a service stops because the service concerned has been removed from the network.
<code>StopByResourceLossEvent</code>	This event is generated whenever presentation of a stream stops because the player has lost so many resources that it cannot continue.
<code>SubtitleAvailableEvent</code>	Report that subtitles are available to be presented having been unavailable.
<code>SubtitleNotAvailableEvent</code>	Inform an application that a subtitle stream has vanished from the network.
<code>SubtitleNotSelectedEvent</code>	Report that subtitles are not now selected.
<code>SubtitleSelectedEvent</code>	Report that subtitles are now selected.

**Class Summary**

<code>VideoFormatEvent</code>	The base class for all other events relating to changes in video format
<code>VideoTransformation</code>	<code>VideoTransformation</code> objects express video transformations, i.e.
<b>Exceptions</b>	
<code>CAException</code>	This exception is thrown when access to a media stream is denied by the CA system.

org.dvb.media

# ActiveFormatDescriptionChangedEvent

## Syntax

```
public class ActiveFormatDescriptionChangedEvent extends VideoFormatEvent
```

```
java.lang.Object
|
+--java.util.EventObject
    |
    +--VideoFormatEvent
        |
        +--org.dvb.media.ActiveFormatDescriptionChangedEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Event signalling that the transmitted active format definition has changed

## Constructors

---

### ActiveFormatDescriptionChangedEvent(Object, int)

```
public ActiveFormatDescriptionChangedEvent(java.lang.Object source, int newFormat)
```

Construct the event

#### Parameters:

`source` - the source of the event

`newFormat` - the new active format description

## Methods

---

### getNewFormat()

```
public int getNewFormat()
```

Get the new active format description

#### Returns:

the new active format description. The value of this is represented by one of the constants from the VideoFormatControl class

# org.dvb.media AspectRatioChangedEvent

## Syntax

```
public class AspectRatioChangedEvent extends VideoFormatEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--VideoFormatEvent
|
+--org.dvb.media.AspectRatioChangedEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Event signalling that the aspect ratio of the transmitted video has changed

## Constructors

---

### AspectRatioChangedEvent(Object, int)

```
public AspectRatioChangedEvent(java.lang.Object source, int newRatio)
```

Construct the event

#### Parameters:

`source` - the source of the event

`newRatio` - the new aspect ratio of the transmitted video

## Methods

---

### getNewRatio()

```
public int getNewRatio()
```

Get the new aspect ratio of the transmitted video

#### Returns:

the new aspect ratio of the video. The value of this is represented by one of the constants from the VideoFormatControl class

org.dvb.media

# BackgroundVideoPresentationControl

## Syntax

```
public interface BackgroundVideoPresentationControl extends VideoPresentationControl
```

## All Superinterfaces:

`javax.media.Control`, `VideoPresentationControl`

## Description

A control to support the setting and querying of the video presentation for background players.

## Methods

---

### **getClosestMatch(VideoTransformation)**

```
public VideoTransformation getClosestMatch(VideoTransformation t)
```

This method takes a video transformation and returns the closest match of that video transformation that can be supported for the currently selected video. If the input video transformation can be supported, then the output video transformation will have the same parameters as the input video transformation. The definition of 'closest match' is implementation dependent.

#### **Parameters:**

`t` - the input video transformation

#### **Returns:**

the closest match to the input video transformation. If the input video transformation is supported, then the input video transformation will be returned (the same instance), otherwise a newly created instance will be returned.

---

### **getVideoTransformation()**

```
public VideoTransformation getVideoTransformation()
```

Return the current video transformation

#### **Returns:**

the video transformation (clipping/scaling/positioning) that is currently used for displaying the video.

---

### **setVideoTransformation(VideoTransformation)**

```
public boolean setVideoTransformation(VideoTransformation t)
```

Sets a new video transformation (clipping/scaling/positioning). If the new video transformation is not supported, then the video transformation will not be changed at all (no best effort attempt is made).

#### **Parameters:**

`t` - the new video transformation

#### **Returns:**

true if the video transformation is supported and has been set, false otherwise.

# org.dvb.media CAException

## Syntax

```
public class CAException extends java.io.IOException
```

```
java.lang.Object
|
+--java.lang.Throwable
   |
   +--java.lang.Exception
      |
      +--java.io.IOException
         |
         +--org.dvb.media.CAException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This exception is thrown when access to a media stream is denied by the CA system. It will typically be thrown by calls to `DataSource.start()` when access to the stream accessed by the `DataSource` is denied.

## Constructors

---

### CAException()

```
public CAException()
```

Constructor without a reason

---

### CAException(String)

```
public CAException(java.lang.String reason)
```

Constructor with a reason

#### Parameters:

`reason` - the reason why access to the stream failed



# org.dvb.media CAStopEvent

## Syntax

```
public class CAStopEvent extends javax.media.StopEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--javax.media.ControllerEvent
|
+--javax.media.TransitionEvent
|
+--javax.media.StopEvent
|
+--org.dvb.media.CAStopEvent
```

## All Implemented Interfaces:

javax.media.MediaEvent, java.io.Serializable

## Description

This event is generated whenever access to a service is withdrawn by the CA system, e.g. at the end of a free preview period. It is not generated when an attempt to construct a Player or DataSource fails due to CA restrictions, or when only some of the presented content is not available or alternate content is presented. Generation of this event informs the application that the Player is no longer presenting any content.

## Constructors

---

### CAStopEvent(Controller)

```
public CAStopEvent(javax.media.Controller source)
```

Construct an event.

#### Parameters:

`source` - the controller which was presenting the service

---

### CAStopEvent(Controller, MediaLocator)

```
public CAStopEvent(javax.media.Controller source, javax.media.MediaLocator stream)
```

Construct an event.

#### Parameters:

`source` - the controller which was presenting the service

`stream` - the URL of the stream from which access has been withdrawn.

## Methods

---

### **getStream()**

```
public javax.media.MediaLocator getStream()
```

This method returns the stream from which access has been withdrawn.

**Returns:**

the stream concerned

# org.dvb.media DFCChangedEvent

## Syntax

```
public class DFCChangedEvent extends VideoFormatEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--VideoFormatEvent
|
+--org.dvb.media.DFCChangedEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Event signalling that the decoder format conversion being used has changed

## Constructors

---

### DFCChangedEvent(Object, int)

```
public DFCChangedEvent(java.lang.Object source, int newDFC)
```

Construct the event

#### Parameters:

`source` - the source of the event

`newDFC` - the new decoder format conversion being used

## Methods

---

### getNewDFC()

```
public int getNewDFC()
```

Get the new decoder format conversion

#### Returns:

the new decoder format conversion. The value of this is represented by one of the constants from the VideoFormatControl class

# org.dvb.media DripFeedDataSource

## Syntax

```
public class DripFeedDataSource extends javax.media.protocol.DataSource
```

```
java.lang.Object
|
+--javax.media.protocol.DataSource
|
+--org.dvb.media.DripFeedDataSource
```

## All Implemented Interfaces:

javax.media.protocol.Controls, javax.media.Duration

## Description

This class allows to create a source for a JMF player to be able to feed the decoder progressively with parts of a clip (e.g. I or P MPEG-2 frame) according to the drip-fed mode format defined in the MHP content format chapter.

To start using the drip-feed mode, the application needs to instantiate a player representing a MPEG-2 video decoder and have its source be a DripFeedDataSource instance.

A DripFeedDataSource instance can be obtained by calling the default constructor of the class.

A player that will be bound to a MPEG-2 video decoder (when realized) can be created with the following special URL (locator): "dripfeed://". It is also possible to use a decoder that was instantiated to play a broadcast MPEG-2 stream.

After having the DripFeedDataSource connected to a Player representing a MPEG-2 video decoder, the following rules applies:

- If the feed method is called when the player is in the "prefetched" state the image will be stored so that when the player goes in the "started" state it will be automatically displayed.
- If the feed method is called when the player is in the "started" mode, the frame shall be displayed immediately. In particular it is not required to feed a second frame to the decoder to display the first frame.
- If the feed method is called when the player is in any other state (or if the DripFeedDataSource is not connected to a player), it will be ignored by the platform implementation.

## Constructors

---

### DripFeedDataSource()

```
public DripFeedDataSource()
```

Constructor. A call to the constructor will throw a security exception if the application is not granted the right to use this mode.

## Methods

---

**connect()**

```
public void connect()
```

This method shall not be used and has no effect. This source is considered as always connected.

**Overrides:**

`javax.media.protocol.DataSource.connect()` in class `javax.media.protocol.DataSource`

**Throws:**

`IOException` - never thrown in this sub-class

---

**disconnect()**

```
public void disconnect()
```

This method shall not be used and has no effect. This source is considered as always connected.

**Overrides:**

`javax.media.protocol.DataSource.disconnect()` in class `javax.media.protocol.DataSource`

---

**feed(byte[])**

```
public void feed(byte[] clip_part)
```

This method allows an application to feed the decoder progressively with parts of a clip (e.g. I or P MPEG-2 frame) according to the drip-fed mode format defined in the MHP content format chapter.

The feed method shall not be called more often than every 500ms. If this rule is not respected, display is not guaranteed.

While in the prefetch state the drip feed data source is only required to correctly process a single invocation of this method where the data consists only of a single I frame. Possible additional invocations while in the prefetch state shall have implementation specific results.

**Parameters:**

`clip_part` - Chunk of bytes compliant with the drip-fed mode format defined in the MHP content format chapter (i.e. one MPEG-2 frame with optional syntactic MPEG-2 elements).

---

**getContentType()**

```
public java.lang.String getContentType()
```

This method shall return the content type for mpeg-2 video "drips"

**Overrides:**

`javax.media.protocol.DataSource.getContentType()` in class `javax.media.protocol.DataSource`

**Returns:**

the content type for MPEG-2 video drips

---

**getControl(String)**

```
public java.lang.Object getControl(java.lang.String controlType)
```

Obtain the object that implements the specified Class or Interface. The full class or interface name must be used. If the control is not supported then null is returned.

**Overrides:**

`javax.media.protocol.DataSource.getControl(java.lang.String)` in class `javax.media.protocol.DataSource`

**Parameters:**

`controlType` - the full class or interface name of the requested control

**Returns:**

the object that implements the control, or null.

---

**getControls()**

```
public java.lang.Object[] getControls()
```

Obtain the collection of objects that control this object. If no controls are supported, a zero length array is returned.

**Overrides:**

`javax.media.protocol.DataSource.getControls()` in class `javax.media.protocol.DataSource`

**Returns:**

the collection of object controls

---

**getDuration()**

```
public javax.media.Time getDuration()
```

This method shall not be used and has no effect.

**Overrides:**

`javax.media.protocol.DataSource.getDuration()` in class `javax.media.protocol.DataSource`

**Returns:**

DURATION\_UNKNOWN.

---

**getStreams()**

```
public javax.media.protocol.PullSourceStream[] getStreams()
```

This method is not used and shall return null.

**Returns:**

always returns null in this sub-class

---

**start()**

```
public void start()
```

This method shall not be used and has no effect. This source is considered as always started.

**Overrides:**

`javax.media.protocol.DataSource.start()` in class `javax.media.protocol.DataSource`

**Throws:**

`IOException` - never thrown in this sub-class

---

**stop()**

```
public void stop()
```

This method shall not be used and has no effect. This source is considered as always started.

**Overrides:**

`javax.media.protocol.DataSource.stop()` in class `javax.media.protocol.DataSource`

**Throws:**

`IOException` - never thrown in this sub-class

# org.dvb.media DripFeedPermission

## Syntax

```
public class DripFeedPermission extends java.security.BasicPermission
```

```
java.lang.Object
|
+--java.security.Permission
   |
   +--java.security.BasicPermission
      |
      +--org.dvb.media.DripFeedPermission
```

## All Implemented Interfaces:

java.security.Guard, java.io.Serializable

## Description

This class represents a permission to access the drip feed mode.

## Constructors

---

### DripFeedPermission(String)

```
public DripFeedPermission(java.lang.String name)
```

Create a new DripFeedPermission.

#### Parameters:

`name` - the name string is currently unused and should be empty

---

### DripFeedPermission(String, String)

```
public DripFeedPermission(java.lang.String name, java.lang.String actions)
```

Create a new DripFeedPermission. This constructor is used by the policy class to instantiate new permission objects.

#### Parameters:

`name` - The name string is currently unused and should be empty

`actions` - The actions string is currently unused and should be null.

org.dvb.media

# NoComponentSelectedEvent

## Syntax

```
public class NoComponentSelectedEvent extends javax.media.StopEvent
```

```
java.lang.Object
|
+--java.util.EventObject
    |
    +--javax.media.ControllerEvent
        |
        +--javax.media.TransitionEvent
            |
            +--javax.media.StopEvent
                |
                +--org.dvb.media.NoComponentSelectedEvent
```

## All Implemented Interfaces:

javax.media.MediaEvent, java.io.Serializable

## Description

This event is generated whenever presentation of a stream stops because there are no selected components to present. One example of this would be use of the `javax.tv.media.MediaSelectControl.remove` method to remove all components of a service. Generation of this event informs the application that the Player is no longer presenting any content.

## Since:

MHP 1.0.1

## Constructors

---

### NoComponentSelectedEvent(Controller)

```
public NoComponentSelectedEvent(javax.media.Controller source)
```

Construct an event.

#### Parameters:

`source` - the controller which was presenting the service

---

### NoComponentSelectedEvent(Controller, MediaLocator)

```
public NoComponentSelectedEvent(javax.media.Controller source,
    javax.media.MediaLocator stream)
```

Construct an event.

#### Parameters:

`source` - the controller which was presenting the service

`stream` - the locator of the stream whose presentation has stopped



## Methods

---

### **getStream()**

```
public javax.media.MediaLocator getStream()
```

This method returns the stream whose presentation has stopped

**Returns:**

the stream concerned

# org.dvb.media PresentationChangedEvent

## Syntax

```
public class PresentationChangedEvent extends javax.media.ControllerEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--javax.media.ControllerEvent
|
+--org.dvb.media.PresentationChangedEvent
```

## All Implemented Interfaces:

javax.media.MediaEvent, java.io.Serializable

## Description

This event is generated whenever the content being presented by a player changes for reasons outside the control of the application. The state of the player does not change - only the content being presented.

## Fields

---

### CA\_FAILURE

```
public static final int CA_FAILURE
```

Presentation changed due an action by the CA subsystem. Alternate content is being played, not the content selected by the user (e.g. adverts in place of a scrambled service)

**See Also:**

[getReason\(\)](#)

---

### CA\_RETURNED

```
public static final int CA_RETURNED
```

Presentation changed due to an action by the CA subsystem. Normal content is now being presented as requested by the user. This reason code is used when the CA subsystem commands the MHP terminal to switch back to the normal presentation after having previously selected an alternate content.

**See Also:**

[getReason\(\)](#)

---

### STREAM\_UNAVAILABLE

```
public static final int STREAM_UNAVAILABLE
```

The stream being presented is no longer available in the transport stream.

**See Also:**

[getReason\(\)](#)

## Constructors

---

### PresentationChangedEvent(Controller, MediaLocator, int)

```
public PresentationChangedEvent(javax.media.Controller source,  
                               javax.media.MediaLocator stream, int reason)
```

Constructor for the event

**Parameters:**

`source` - the controller whose presentation changed

`stream` - the stream now being presented.

`reason` - the reason for the change encoded as one of the constants in this class

## Methods

---

### getReason()

```
public int getReason()
```

This method returns the reason why access has been withdrawn.

**Returns:**

the reason for the change specified when the event was constructed

---

### getStream()

```
public javax.media.MediaLocator getStream()
```

This method returns the locator for the stream now being presented.

**Returns:**

the stream now being presented

# org.dvb.media ServiceRemovedEvent

## Syntax

```
public class ServiceRemovedEvent extends javax.media.StopEvent
```

```

java.lang.Object
|
+--java.util.EventObject
    |
    +--javax.media.ControllerEvent
        |
        +--javax.media.TransitionEvent
            |
            +--javax.media.StopEvent
                |
                +--org.dvb.media.ServiceRemovedEvent

```

## All Implemented Interfaces:

javax.media.MediaEvent, java.io.Serializable

## Description

This event is generated whenever access to a service stops because the service concerned has been removed from the network. Generation of this event informs the application that the Player is no longer presenting any content.

## Since:

MHP 1.0.1

## Constructors

---

### ServiceRemovedEvent(Controller)

```
public ServiceRemovedEvent(javax.media.Controller source)
```

Construct an event.

#### Parameters:

`source` - the controller which was presenting the service

---

### ServiceRemovedEvent(Controller, MediaLocator)

```
public ServiceRemovedEvent(javax.media.Controller source, javax.media.MediaLocator stream)
```

Construct an event.

#### Parameters:

`source` - the controller which was presenting the service

`stream` - the locator of the stream which was removed from the network

## Methods

---

**getStream()**

```
public javax.media.MediaLocator getStream()
```

This method returns the stream which was removed from the network

**Returns:**

the stream concerned

# org.dvb.media StopByResourceLossEvent

## Syntax

```
public class StopByResourceLossEvent extends javax.media.StopEvent
```

```

java.lang.Object
|
+--java.util.EventObject
    |
    +--javax.media.ControllerEvent
        |
        +--javax.media.TransitionEvent
            |
            +--javax.media.StopEvent
                |
                +--org.dvb.media.StopByResourceLossEvent

```

## All Implemented Interfaces:

javax.media.MediaEvent, java.io.Serializable

## Description

This event is generated whenever presentation of a stream stops because the player has lost so many resources that it cannot continue. Generation of this event informs the application that the Player is no longer presenting any content.

## Since:

MHP 1.0.1

## Constructors

---

### StopByResourceLossEvent(Controller)

```
public StopByResourceLossEvent(javax.media.Controller source)
```

Construct an event.

#### Parameters:

`source` - the controller which was presenting the service

---

### StopByResourceLossEvent(Controller, MediaLocator)

```
public StopByResourceLossEvent(javax.media.Controller source,
    javax.media.MediaLocator stream)
```

Construct an event.

#### Parameters:

`source` - the controller which was presenting the service

`stream` - the locator of the stream which was being presented

## Methods

---

### **getStream()**

```
public javax.media.MediaLocator getStream()
```

This method returns the stream which was being presented

**Returns:**

the stream concerned

# org.dvb.media SubtitleAvailableEvent

## Syntax

```
public class SubtitleAvailableEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.dvb.media.SubtitleAvailableEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Report that subtitles are available to be presented having been unavailable. This event is not generated on service selection or other forms of 'zapping'. Its generation is restricted to changes in the composition of the subtitle aspects of the same broadcast stream.

## Constructors

---

### SubtitleAvailableEvent(SubtitleLanguageControl)

```
public SubtitleAvailableEvent(org.davic.media.SubtitlingLanguageControl source)
```

Constructor.

#### Parameters:

source - the source of the event

## Methods

---

### getSource()

```
public java.lang.Object getSource()
```

Return the control which was the source of the event.

#### Overrides:

java.util.EventObject.getSource() in class java.util.EventObject

#### Returns:

the source of the event



# org.dvb.media SubtitleListener

## Syntax

```
public interface SubtitleListener extends java.util.EventListener
```

## All Superinterfaces:

java.util.EventListener

## Description

Report that a subtitle event has happened.

## Methods

---

### subtitleStatusChanged(EventObject)

```
public void subtitleStatusChanged(java.util.EventObject event)
```

Report a subtitle event has happened.

#### Parameters:

`event` - the event which happened

# org.dvb.media SubtitleNotAvailableEvent

## Syntax

```
public class SubtitleNotAvailableEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.dvb.media.SubtitleNotAvailableEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Inform an application that a subtitle stream has vanished from the network. This event is not generated on service selection or other forms of 'zapping'. Its generation is restricted to changes in the composition of the subtitle aspects of the same broadcast stream.

## Constructors

---

### SubtitleNotAvailableEvent(SubtitleLanguageControl)

```
public SubtitleNotAvailableEvent(org.davic.media.SubtitlingLanguageControl source)
```

Constructor.

#### Parameters:

source - the source of the event

## Methods

---

### getSource()

```
public java.lang.Object getSource()
```

Return the source of the event.

#### Overrides:

java.util.EventObject.getSource() in class java.util.EventObject

#### Returns:

the source of the event

# org.dvb.media SubtitleNotSelectedEvent

## Syntax

```
public class SubtitleNotSelectedEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.dvb.media.SubtitleNotSelectedEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Report that subtitles are not now selected. Even if subtitles are available in the network, they will not be presented. This event is generated when the combination of end user control of subtitles through the navigator and application control of subtitles through `SubtitlingLanguageControl.setSubtitling` changes whether subtitles are to be presented if they are available. It is not generated for changes in the underlying availability of subtitles even if those cause changes in whether subtitles are presented or not.

## Constructors

---

### SubtitleNotSelectedEvent(SubtitlingLanguageControl)

```
public SubtitleNotSelectedEvent(org.davic.media.SubtitlingLanguageControl source)
```

Constructor

#### Parameters:

source - the source of the event

## Methods

---

### getSource()

```
public java.lang.Object getSource()
```

Return the source of the event

#### Overrides:

java.util.EventObject.getSource() in class java.util.EventObject

#### Returns:

the source of the event

# org.dvb.media SubtitleSelectedEvent

## Syntax

```
public class SubtitleSelectedEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.dvb.media.SubtitleSelectedEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Report that subtitles are now selected. If subtitles are also available then they will be presented. This event is generated when the combination of end user control of subtitles through the navigator and application control of subtitles through `SubtitlingLanguageControl.setSubtitling` changes whether subtitles are to be presented if they are available. It is not generated for changes in the underlying availability of subtitles even if those cause changes in whether subtitles are presented or not.

## Constructors

---

### SubtitleSelectedEvent(SubtitlingLanguageControl)

```
public SubtitleSelectedEvent(org.davic.media.SubtitlingLanguageControl source)
```

Constructor

#### Parameters:

`source` - the source of the event

## Methods

---

### getSource()

```
public java.lang.Object getSource()
```

Return the source of the event

#### Overrides:

`java.util.EventObject.getSource()` in class `java.util.EventObject`

#### Returns:

the source of the event

# org.dvb.media SubtitlingEventControl

## Syntax

```
public interface SubtitlingEventControl extends org.davic.media.SubtitlingLanguageControl
```

## All Superinterfaces:

javax.media.Control, org.davic.media.LanguageControl, org.davic.media.SubtitlingLanguageControl

## Description

Allow applications to register and unregister their interest in events related to the availability and presentation of subtitles.

## Methods

---

### addSubtitleListener(SubtitleListener)

```
public void addSubtitleListener(SubtitleListener l)
```

Add a listener for subtitle events

#### Parameters:

l - the listener to report the events to

---

### removeSubtitleListener(SubtitleListener)

```
public void removeSubtitleListener(SubtitleListener l)
```

Remove a listener for subtitle events

#### Parameters:

l - the listener to remove

# org.dvb.media VideoFormatControl

## Syntax

```
public interface VideoFormatControl extends javax.media.Control
```

## All Superinterfaces:

javax.media.Control

## Description

This provides a means for applications to get information associated with the format and aspect ratio of the video being presented to the user. This control will only be available for Players presenting MPEG-2 video streams.

It is important to note that due to different video and display formats (and user preferences), not all of the full video frame may be displayed. Similarly, it may not always be possible to map video and graphics with perfect accuracy.

## Fields

---

### AFD\_14\_9

```
public static final int AFD_14_9
```

Constant representing an MPEG active format description of 14:9 (centre)

---

### AFD\_14\_9\_TOP

```
public static final int AFD_14_9_TOP
```

Constant representing an MPEG active format description of 14:9 (top)

---

### AFD\_16\_9

```
public static final int AFD_16_9
```

Constant representing an MPEG active format description of 16:9 (centre)

---

### AFD\_16\_9\_SP\_14\_9

```
public static final int AFD_16_9_SP_14_9
```

Constant representing an MPEG active format description of 16:9 (with shoot & protect 14:9 centre)

---

### AFD\_16\_9\_SP\_4\_3

```
public static final int AFD_16_9_SP_4_3
```

Constant representing an MPEG active format description of 16:9 (with shoot & protect 4:3 centre)

---

**AFD\_16\_9\_TOP**

```
public static final int AFD_16_9_TOP
```

Constant representing an MPEG active format description of 16:9 (top)

---

**AFD\_4\_3**

```
public static final int AFD_4_3
```

Constant representing an MPEG active format description of 4:3 (centre)

---

**AFD\_4\_3\_SP\_14\_9**

```
public static final int AFD_4_3_SP_14_9
```

Constant representing an MPEG active format description of 4:3 (with shoot & protect 14:9 centre)

---

**AFD\_GT\_16\_9**

```
public static final int AFD_GT_16_9
```

Constant representing an MPEG active format description of greater than 16:9 (centre)

---

**AFD\_NOT\_PRESENT**

```
public static final int AFD_NOT_PRESENT
```

Constant showing an MPEG active format description is not present

---

**AFD\_SAME**

```
public static final int AFD_SAME
```

Constant representing an MPEG active format description that is the same as the coded frame

---

**ASPECT\_RATIO\_16\_9**

```
public static final int ASPECT_RATIO_16_9
```

Constant representing an aspect ratio of 16:9

---

**ASPECT\_RATIO\_2\_21\_1**

```
public static final int ASPECT_RATIO_2_21_1
```

Constant representing an aspect ratio of 2.21:1

---

**ASPECT\_RATIO\_4\_3**

```
public static final int ASPECT_RATIO_4_3
```

Constant representing an aspect ratio of 4:3

---

**ASPECT\_RATIO\_UNKNOWN**

```
public static final int ASPECT_RATIO_UNKNOWN
```

Constant representing an unknown aspect ratio

---

**DAR\_16\_9**

```
public static final int DAR_16_9
```

Constant representing a display aspect ratio of 16:9

---

**DAR\_4\_3**

```
public static final int DAR_4_3
```

Constant representing a display aspect ratio of 4:3

---

**DFC\_PLATFORM**

```
public static final int DFC_PLATFORM
```

Control over the decoder format conversions is returned to being managed by the platform. This is the same as the value used if no MHP application has set a video transformation. It is not required to correspond to a single decoder format conversion and may change over time as the video input format & signalling change. This constant can only be used to set the decoder format conversion. Reading the decoder format conversion shall always return the DFC used at the time concerned.

---

**DFC\_PROCESSING\_CCO**

```
public static final int DFC_PROCESSING_CCO
```

A 4:3 central part out of the 720x576 input 16:9 frame is transferred into a 720x576 4:3 output frame

---

**DFC\_PROCESSING\_FULL**

```
public static final int DFC_PROCESSING_FULL
```

The full 720x576 frame is transferred (this may be either 4:3 or 16:9; part of this may be black, e.g. in the "pillar box" cases)

---

**DFC\_PROCESSING\_LB\_14\_9**

```
public static final int DFC_PROCESSING_LB_14_9
```

The 720x576 input grid is transferred into a 14:9 LB in a 4:3 frame

---

**DFC\_PROCESSING\_LB\_16\_9**

```
public static final int DFC_PROCESSING_LB_16_9
```

The 720x576 input grid is transferred into a 16:9 letterbox in a 4:3 frame

---

**DFC\_PROCESSING\_LB\_2\_21\_1\_ON\_16\_9**

```
public static final int DFC_PROCESSING_LB_2_21_1_ON_16_9
```

The 720x576 input grid is transferred into a 2.21:1 letterbox in a 16:9 frame.



---

**DFC\_PROCESSING\_LB\_2\_21\_1\_ON\_4\_3**

```
public static final int DFC_PROCESSING_LB_2_21_1_ON_4_3
```

The 720x576 input grid is transferred into a 2.21:1 letterbox in a 4:3 frame.

---

**DFC\_PROCESSING\_NONE**

```
public static final int DFC_PROCESSING_NONE
```

Decoder format conversion is inactive

---

**DFC\_PROCESSING\_PAN\_SCAN**

```
public static final int DFC_PROCESSING_PAN_SCAN
```

A 4:3 part out of the 720x576 input 16:9 or 2.21:1 frame is transferred into a 720x576 4:3 output frame. The horizontal position of this part is determined by pan&scan vectors from the MPEG video stream.

---

**DFC\_PROCESSING\_UNKNOWN**

```
public static final int DFC_PROCESSING_UNKNOWN
```

Constant representing an unknown format conversion being performed by the decoder

## Methods

---

**addVideoFormatListener(VideoFormatListener)**

```
public void addVideoFormatListener(VideoFormatListener l)
```

Add a listener for VideoFormatChangedEvents

**Parameters:**

l - the listener to add

---

**getActiveFormatDefinition()**

```
public int getActiveFormatDefinition()
```

Return the value of the active\_format field of the MPEG Active Format Description of the video if it is transmitted (one of the constants AFD\_\* above). If this field is not available then AFD\_NOT\_PRESENT is returned. The constant values for the constants representing the Active Format Description should be identical to the values specified in ETR154, annex B.

**Returns:**

the value of the active\_format field of the MPEG Active Format Description of the video if it is transmitted. If this field is not available, or the video is not MPEG, then AFD\_NOT\_PRESENT is returned.

---

**getAspectRatio()**

```
public int getAspectRatio()
```

Return the aspect ratio of the video as it is transmitted. If the aspect ratio is not known, ASPECT\_RATIO\_UNKNOWN is returned

**Returns:**

the aspect ratio of the video

---

**getDecoderFormatConversion()**

```
public int getDecoderFormatConversion()
```

Return a value representing what format conversion is being done by the decoder in the platform (one of the constants DFC\_\* above). A receiver may implement only a subset of the available options. This decoder format conversion may be active or not depending upon the mode of operation.

**Returns:**

the decoder format conversion being performed or DFC\_PROCESSING\_UNKNOWN if this is not known

---

**getDisplayAspectRatio()**

```
public int getDisplayAspectRatio()
```

Return the aspect ratio of the display device connected to this MHP decoder (one of the constants DAR\_\* above)

**Returns:**

the aspect ratio of the display device connected to the decoder

---

**getVideoTransformation(int)**

```
public VideoTransformation getVideoTransformation(int dfc)
```

This method returns a VideoTransformation object that corresponds with the specified Decoder Format Conversion when applied to the currently selected video. If the specified Decoder Format Conversion is not supported for the currently selected video, then this method returns null.

**Parameters:**

dfc - the Decoder Format Conversion (one of the DFC\_\* constants specified in this interface)

**Returns:**

the video transformation, or null if the specified Decoder Format Conversion is not supported for the currently selected video.

---

**removeVideoFormatListener(VideoFormatListener)**

```
public void removeVideoFormatListener(VideoFormatListener l)
```

Remove a listener for VideoFormatChangedEvents

**Parameters:**

l - the listener to remove

# org.dvb.media VideoFormatEvent

## Syntax

```
public abstract class VideoFormatEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.dvb.media.VideoFormatEvent
```

## Direct Known Subclasses:

`ActiveFormatDescriptionChangedEvent`, `AspectRatioChangedEvent`,  
`DFCChangedEvent`

## All Implemented Interfaces:

`java.io.Serializable`

## Description

The base class for all other events relating to changes in video format

## Constructors

---

### VideoFormatEvent(Object)

```
public VideoFormatEvent(java.lang.Object source)
```

Constructor

#### Parameters:

`source` - the source of the event

org.dvb.media

# VideoFormatListener

## Syntax

```
public interface VideoFormatListener
```

## Description

The listener used to receive video format events

## Methods

---

### receiveVideoFormatEvent(VideoFormatEvent)

```
public void receiveVideoFormatEvent(VideoFormatEvent anEvent)
```

receive a VideoFormatEvent

#### Parameters:

`anEvent` - the VideoFormatEvent that has been received

org.dvb.media

# VideoPresentationControl

## Syntax

```
public interface VideoPresentationControl extends javax.media.Control
```

## All Known Subinterfaces:

[BackgroundVideoPresentationControl](#)

## All Superinterfaces:

[javax.media.Control](#)

## Description

A control to support setting and querying the video presentation.

Note: For a component-based player the scaling and positioning of the video is done by manipulating the corresponding AWT component. The VideoPresentationControl only allows for the setting of the clipping region.

Note: If the hardware supports the positioning of interlaced video on even lines only (when counting from 0), then a component-based player is allowed to position the top of the video one line below where it should be.

For a background player there is the BackgroundVideoPresentationControl that allows for the setting of the clipping region, the position and the scaling of the video in one atomic action.

## Fields

---

### POS\_CAP\_FULL

```
public static final byte POS_CAP_FULL
```

Constant representing that the video can be positioned anywhere on the screen, even if a part of the video is off screen as a result of that.

---

### POS\_CAP\_FULL\_EVEN\_LINES

```
public static final byte POS_CAP_FULL_EVEN_LINES
```

n Constant representing that the video can be positioned anywhere on the screen, even if a part of the video is off screen as a result of that, with the restriction that the field order is respected. This implies that interlaced video can be positioned on even lines only (when counting from 0).

---

### POS\_CAP\_FULL\_EVEN\_LINES\_IF\_ENTIRE\_VIDEO\_ON\_SCREEN

```
public static final byte POS_CAP_FULL_EVEN_LINES_IF_ENTIRE_VIDEO_ON_SCREEN
```

Constant representing that the video can be positioned anywhere on screen as long as all the video is on screen, with the restriction that the field order is respected. This implies that interlaced video can be positioned on even lines only (when counting from 0).

---

**POS\_CAP\_FULL\_IF\_ENTIRE\_VIDEO\_ON\_SCREEN**

```
public static final byte POS_CAP_FULL_IF_ENTIRE_VIDEO_ON_SCREEN
```

Constant representing that the video can be positioned anywhere on screen as long as all the video is on screen.

---

**POS\_CAP\_OTHER**

```
public static final byte POS_CAP_OTHER
```

Constant representing that the video positioning capability cannot be expressed by another POS\_CAP\_\* constant.

## Methods

---

**getActiveVideoArea()**

```
public org.havi.ui.HScreenRectangle getActiveVideoArea()
```

This method returns the size and location of the active video area. The active video area excludes any "bars" used for letterboxing or pillarboxing that the receiver knows about. Bars that are included in the broadcast stream and not signalled by active format descriptors are included in the active video area. The active video area may be larger/smaller than the screen, and may possibly be offset. The offsets will be negative if the origin of the active video area is above/left of the top, left corner of the screen. In case of pan&scan, the value returned may vary over time. This method only describes the relationship between the active video and the screen. It does not describe which portion of the screen is displaying the video.

Note: This method includes any video scaling.

**Returns:**

an HScreenRectangle representing the active video area in the normalised coordinate space.

---

**getActiveVideoAreaOnScreen()**

```
public org.havi.ui.HScreenRectangle getActiveVideoAreaOnScreen()
```

This method returns the size and location of the active video area on-screen. The active video area excludes any "bars" used for letterboxing or pillarboxing that the receiver knows about. Bars that are included in the broadcast stream and not signalled by active format descriptors are included in the active video area. The active video area on-screen may be smaller than the area of the screen, and may possibly be offset a positive amount. This method only describes the area on-screen where active video is being presented. It does not really describe which part of the video is being shown on-screen. This is especially true for pan&scan.

Note: This method includes any video scaling.

**Returns:**

an HScreenRectangle representing the active video area on-screen in the normalised coordinate space.

---

**getClipRegion()**

```
public java.awt.Rectangle getClipRegion()
```

This method returns the area of the decoded video that will be displayed. If clipping is not supported, the dimensions of the bounding box will be the same as the displayed video.

**Returns:**

area of the decoded video that will be displayed. The coordinate space used to express the region is that of the decoded video after possible ETR154 up-sampling.

---

**getHorizontalScalingFactors()**

```
public float[] getHorizontalScalingFactors()
```

This method gives information about the supported discrete horizontal scaling factors in case arbitrary horizontal scaling is not supported.

**Returns:**

an array with the supported discrete horizontal scaling factors (including the scaling factor 1), sorted in ascending order. null is returned when arbitrary horizontal scaling is supported.

---

**getInputVideoSize()**

```
public java.awt.Dimension getInputVideoSize()
```

This method returns the dimensions of the video before any scaling has taken place (but after ETR154 up-sampling). On 50Hz standard definition systems this method always returns 720x576.

**Returns:**

the size of the decoded video before any scaling has taken place (but after ETR154 up-sampling)

---

**getPositioningCapability()**

```
public byte getPositioningCapability()
```

This method gives information about how the video can be positioned on screen.

**Returns:**

the positioning capability for the currently selected video as one of the POS\_CAP\_\* constants.

---

**getTotalVideoArea()**

```
public org.havi.ui.HScreenRectangle getTotalVideoArea()
```

This method returns a relative size and location of the total video area, including any "bars" used for letterboxing or pillarboxing that are included in the broadcast stream, but excluding any "bars" introduced as a result of video filtering. This may be larger or smaller than the size of the physical display device. This method only describes the relationship between the total video and the screen. It does not describe which portion of the screen is displaying the video.

Note: This method includes any video scaling.

**Returns:**

an HScreenRectangle representing the total video area in the normalised coordinate space.

---

**getTotalVideoAreaOnScreen()**

```
public org.havi.ui.HScreenRectangle getTotalVideoAreaOnScreen()
```

This method returns a relative size and location of the total video area on-screen, including any "bars" used for letterboxing or pillarboxing that are included in the broadcast stream, but excluding any "bars" introduced as a result of video filtering. This method only describes the area on-screen

where total video is being presented. This does not really describe which part of the video is being shown on-screen. This is especially true for pan&scan.

Note: This method includes any video scaling.

**Returns:**

an HScreenRectangle representing the total video area on-screen in the normalised coordinate space.

---

### getVerticalScalingFactors()

```
public float[] getVerticalScalingFactors()
```

This method gives information about the supported discrete vertical scaling factors in case arbitrary vertical scaling is not supported.

**Returns:**

an array with the supported discrete vertical scaling factors (including the scaling factor 1), sorted in ascending order. null is returned when arbitrary vertical scaling is supported.

---

### getVideoSize()

```
public java.awt.Dimension getVideoSize()
```

This method returns the size of the decoded video as it is being presented to the user. It takes scaling and clipping into account.

**Returns:**

the size of the decoded video as it is being presented to the user

---

### setClipRegion(Rectangle)

```
public java.awt.Rectangle setClipRegion(java.awt.Rectangle clipRect)
```

Set the region of the decoded video that will be displayed. If clipping is not supported, this method has no effect. If the bounding box extends beyond the decoded video, the area not containing video will be filled with the background. By default, the clipping region is set to the dimensions of the decoded video. This method returns the bounding box of the clipping region that was actually set. Implementations may approximate the requested rectangle if they have restrictions on video clipping.

If the player is a component-based player (as opposed to a background player), then the top left corner of the clip region will be aligned with the top left corner of the `java.awt.Component` returned by the method `javax.media.Player.getVisualComponent()`. Hence changing the position of the clip region within the video moves the video with respect to the coordinate space used by `java.awt`.

**Parameters:**

`clipRect` - the bounding box of the clipping region. The coordinate space used to express the region is that of the decoded video after possible ETR154 up-sampling.

**Returns:**

the set clipping region. If the requested clipping region is supported exactly, then the input parameter `clipRect` is returned, otherwise a newly created object will be returned.

---

### supportsArbitraryHorizontalScaling()

```
public float[] supportsArbitraryHorizontalScaling()
```

This method gives information about whether arbitrary horizontal scaling is supported for the currently playing video. If arbitrary horizontal scaling is supported, then an array with two elements is returned. The first element returns the smallest allowed scaling factor (e.g. 0.5) and the second element returns the largest allowed scaling factor (e.g. 4). If arbitrary horizontal scaling is not



supported, null is returned. In that case the method `getHorizontalScalingFactors` can be used to query which discrete scaling factors are supported.

**Returns:**

an array with the minimum and maximum allowed horizontal scaling factor, or null if arbitrary horizontal scaling is not supported.

---

**supportsArbitraryVerticalScaling()**

```
public float[] supportsArbitraryVerticalScaling()
```

This method gives information about whether arbitrary vertical scaling is supported for the currently playing video. If arbitrary vertical scaling is supported, then an array with two elements is returned. The first element returns the smallest allowed scaling factor (e.g. 0.5) and the second element returns the largest allowed scaling factor (e.g. 2). If arbitrary vertical scaling is not supported, null is returned. In that case the method `getVerticalScalingFactors` can be used to query which discrete scaling factors are supported.

**Returns:**

an array with the minimum and maximum allowed vertical scaling factor, or null if arbitrary vertical scaling is not supported.

---

**supportsClipping()**

```
public boolean supportsClipping()
```

Test if the decoder supports clipping

**Returns:**

true if and only if the decoder supports displaying only a section of the decoded video.

# org.dvb.media VideoTransformation

## Syntax

```
public class VideoTransformation

java.lang.Object
|
+--org.dvb.media.VideoTransformation
```

## Description

VideoTransformation objects express video transformations, i.e. the clipping, the horizontal and vertical scaling and the position of the video. All transformations are to be applied after possible ETR154 up-sampling.

Note: Instances of VideoTransformation can represent pan and scan, but an application cannot create such instances itself. An application can get a VideoTransformation representing pan and scan, by calling the VideoFormatControl.getVideoTransformation() method with the pan and scan Decoder Format Conversion constant.

## Constructors

---

### VideoTransformation()

```
public VideoTransformation()
```

Creates a VideoTransformation object with default parameters. Clipping is disabled, both the horizontal and the vertical scaling factors are 1, and the video position is (0, 0) in the normalised coordinate space.

---

### VideoTransformation(Rectangle, float, float, HScreenPoint)

```
public VideoTransformation(java.awt.Rectangle clipRect, float horizontalScalingFactor,
    float verticalScalingFactor, org.havi.ui.HScreenPoint location)
```

Creates a VideoTransformation object with the supplied parameters.

#### Parameters:

`clipRect` - the bounding box of the clipping region. The coordinate space used to express the region is that of the decoded video after possible ETR154 up-sampling.

`horizontalScalingFactor` - the horizontal scaling factor.

`verticalScalingFactor` - the vertical scaling factor.

`location` - the location of the video on the screen in the normalised coordinate space.

## Methods

---

### getClipRegion()

```
public java.awt.Rectangle getClipRegion()
```

Gets the clipping region.

**Returns:**

the bounding box of the clipping region. The coordinate space used to express the region is that of the decoded video after possible ETR154 up-sampling. null is returned if this video transformation represents pan and scan.

---

**getScalingFactors()**

```
public float[] getScalingFactors()
```

Gets the horizontal and vertical scaling factors.

**Returns:**

an array with two elements. The first element contains the horizontal scaling factor, the second element the vertical scaling factor.

---

**getVideoPosition()**

```
public org.havi.ui.HScreenPoint getVideoPosition()
```

Returns the video position.

**Returns:**

the location of the video on the screen in the normalised coordinate space.

---

**isPanAndScan()**

```
public boolean isPanAndScan()
```

Returns whether this video transformation represents pan and scan.

**Returns:**

true is this video transformation represents pan and scan, false otherwise.

---

**setClipRegion(Rectangle)**

```
public void setClipRegion(java.awt.Rectangle clipRect)
```

Sets the clipping region.

If this video transformation represents pan and scan, then it will no longer represent pan and scan when this method is called.

**Parameters:**

`clipRect` - the bounding box of the clipping region. The coordinate space used to express the region is that of the decoded video after possible ETR154 up-sampling.

---

**setScalingFactors(float, float)**

```
public void setScalingFactors(float horizontalScalingFactor, float verticalScalingFactor)
```

Sets the horizontal and vertical scaling factors.

**Parameters:**

`horizontalScalingFactor` - the horizontal scaling factor.

`verticalScalingFactor` - the vertical scaling factor.

---

**setVideoPosition(HScreenPoint)**

```
public void setVideoPosition(org.havi.ui.HScreenPoint location)
```

Sets the video position.

**Parameters:**

`location` - the location of the video on the screen in the normalised coordinate space.

---

# Annex O (normative): Integration of the JavaTV SI API and DVB SI

## O.1 Introduction

This section describes how the JavaTV Service Information API as described in [51] can be mapped to the data structures of DVB Service Information as defined in EN 300 468 [4]. Secondly this document describes how the JavaTV API and the DVB SI API (as described in annex M, "(normative): SI Access API" on page 490) can be integrated.

## O.2 Mapping of the JavaTV SI API to DVB SI

This section describes for every relevant Java interface and method in the JavaTV SI API how it is mapped to the DVB Service Information.

### O.2.1 javax.tv.service.Service

The Service interface represents a DVB Service as stored in the MHP terminal as "installed services".

Depending on the MHP terminal implementation, the objects implementing this interface may or may not implement the ServiceNumber interface as well. Furthermore, it is allowed that even within the same MHP terminal implementation, some objects implementing the Service interface implement the ServiceNumber while other objects implement only the Service interface.

Objects implementing this interfaces and representing DVB services shall also implement the `org.dvb.si.TextualServiceIdentifiersQuery` interface.

The methods are mapped as follows:

#### O.2.1.1 getName

Returns the name of the service as stored in the MHP terminal. Depending on the MHP terminal implementation, the end user may have the possibility to edit these names according to his preferences. If the contents of this field are retrieved by the MHP terminal by default from DVB SI, it is recommended that the MHP terminal uses the abbreviated form of the service name from the Service descriptor.

#### O.2.1.2 getServiceType

Returns the ServiceType according to the mapping defined in section O.2.3, "javax.tv.service.ServiceType" on page 618.

### O.2.2 javax.tv.service.ServiceComponent

The ServiceComponent interface provides the information contained in the Component descriptors, Multilingual component descriptors or Data broadcast descriptors in the EIT.

#### O.2.2.1 getComponentName

Returns the component description from the Component descriptor or from the Multilingual component descriptor using the language preference, or from the Data broadcast descriptor.

#### O.2.2.2 getAssociatedLanguage

Returns the ISO 639.2 [66] language code indicating the language of the component (i.e. not necessarily the selected language for the name returned by `getName()`) from the Component descriptor or from the Data broadcast descriptor.

#### O.2.2.3 getStreamType

Returns the stream type according to the mapping from the `stream_content` field and the `component_type` field of the Component descriptor or the Data broadcast descriptor to the JavaTV stream types according to O.2.4, "javax.tv.service.StreamType" on page 618.

### O.2.3 javax.tv.service.ServiceType

The DVB SI service types are defined in Table 61 of EN 300 468 [4]. These should be mapped to the JavaTV service types as follows.

**Table O.1 : Mapping DVB to JavaTV service types**

DVB Service type code	DVB Service Type Description	JavaTV Service Type
0x01	Digital television service	DIGITAL_TV
0x02	Digital radio sound service	DIGITAL_RADIO
0x03	Teletext service	DATA_BROADCAST
0x04	NVOD Reference service	NVOD_REFERENCE
0x05	NVOD time-shifted service	NVOD_TIME_SHIFTED
0x06	Mosaic service	DIGITAL_TV
0x07	PAL coded signal	ANALOG_TV
0x08	SECAM coded signal	ANALOG_TV
0x09	D/D2-MAC	ANALOG_TV
0x0A	FM Radio	ANALOG_RADIO
0x0B	NTSC coded signal	ANALOG_TV
0x0C	Data broadcast service	DATA_BROADCAST
0x10	MHP application service	DATA_APPLICATION
0x00, 0x0D...0x0F, 0x11...0xFF		UNKNOWN

### O.2.4 javax.tv.service.StreamType

The DVB SI stream\_content and component\_type values are defined in Table 15 of EN 300 468 [4]. These should be mapped to the JavaTV Stream types as follows. If the component does not have an associated Component descriptor, but a Data broadcast descriptor, the stream type DATA shall be used.

**Table O.2 : Mapping DVB stream & component types to JavaTV**

DVB stream_content	DVB component_type	JavaTV Stream type
0x01	0x00...0xff	VIDEO
0x02	0x00...0xff	AUDIO
0x03	0x01, 0x10...0x13, 0x20...0x23	SUBTITLES
0x03	0x02	DATA
0x03	0x00, 0x14...0x1F, 0x23...0xFF	UNKNOWN
0x04...0x0F	0x00...0xFF	UNKNOWN

### O.2.5 javax.tv.service.SIElement

This interface is implemented by objects implementing the Network, Bouquet, TransportStream, ServiceDetails, ServiceComponent and ProgramEvent interfaces.

#### O.2.5.1 getServiceInformationType

This method shall return the DVB\_SI ServiceInformationType.

## O.2.6 `javax.tv.service.SIManager`

### O.2.6.1 `getSupportedDimensions`

The parental rating descriptor defined in DVB SI standardizes one rating scheme that is based on age. To describe this DVB defined rating scheme, the `getSupportedDimensions` shall return an array that contains the string "DVB Age based rating".

### O.2.6.2 `getRatingDimension`

When given the string "DVB Age based rating", this method shall return an object implementing the `RatingDimension` interface as described in section O.2.10, "`javax.tv.service.navigation.RatingDimension`" on page 620.

### O.2.6.3 `retrieveSIElement`

When passed a locator that points to a service, an object implementing the `ServiceDetails` interface shall be returned. Other types of locators are not supported.

### O.2.6.4 `getTransports`

The object returned by this method shall implement the `Transport` interface as described in O.2.12, "`javax.tv.service.transport.Transport`" on page 620.

### O.2.6.5 `filterServices`

Filtering of Services shall be supported with `ServiceFilters`. The `SIElementFilter` is required to be supported as defined in O.2.7, "`javax.tv.service.navigation.SIElementFilter`" on page 619

## O.2.7 `javax.tv.service.navigation.SIElementFilter`

The `SIElementFilter` allows filtering of Services based on another `SIElement`. This filter type shall be supported for the `Network` and `TransportStream` objects. For other `SIElement` objects, the constructor may throw `FilterNotSupportedException`.

## O.2.8 `javax.tv.service.navigation.ServiceDetails`

The `ServiceDetails` interface represents the information regarding the service as retrieved from the broadcast DVB SI. The object implementing this interface for DVB SI implements the `CAIdentification` interface according to the mapping defined in section O.2.9, "`javax.tv.service.navigation.CAIdentification`" on page 620. These objects shall not implement the `ServiceNumber` interface.

### O.2.8.1 `getLongName`

Returns the full name of the service from the Service descriptor or from the Multilingual service name descriptor using the language preference.

### O.2.8.2 `getServiceType`

Returns the `ServiceType` according to the mapping defined in section O.2.3, "`javax.tv.service.ServiceType`" on page 618.

### O.2.8.3 `retrieveServiceDescription`

Shall always result in a `notifyFailure` of the `SIREquestor` object being called with the `DATA_UNAVAILABLE` `SIREquestFailureType`, as DVB SI does not include a service description.

### O.2.8.4 `retrieveComponents`

The information for the `ServiceComponents` shall be retrieved from `Component` descriptors (or their multilingual variants or the `Data` broadcast descriptor for data components) of the present event in the EIT present/following table.

## O.2.9 javax.tv.service.navigation.CAIdentification

This interface shall be implemented by objects implementing the ServiceDetails, ProgramEvent or Bouquet interface.

### O.2.9.1 getCASystemIds

Returns the array of integer values containing the CA\_system\_ids from the CA identifier descriptor. If the CA identifier descriptor is not present, returns an empty array.

### O.2.9.2 isFree

When implemented in an object implementing the ServiceDetails or ProgramEvent interface, this method shall return true if and only if the free\_CA\_mode bit is set to "0" in the SDT or EIT entry, respectively.

When implemented in an object implementing the Bouquet interface, this method shall return true if and only if there is no CA identifier descriptor present in the BAT.

## O.2.10 javax.tv.service.navigation.RatingDimension

The Parental rating descriptor defined in DVB SI standardizes one rating scheme that is based on age. This rating scheme contains 15 distinct age rating levels from 4 to 18 years.

An object that describes this DVB defined rating scheme shall implement the methods as follows.

### O.2.10.1 getDimensionName

Returns the string "DVB Age based rating".

### O.2.10.2 getNumberOfLevels

Returns 15.

### O.2.10.3 getRatingLevelDescription

Returns an array of 2 strings of the form:

```
{"Over n", "Recommended minimum age: n years"}
```

where n is the input parameter + 4.

## O.2.11 javax.tv.service.navigation.ServiceProviderInformation

This interface shall be implemented by objects implementing the ServiceDetails interface.

### O.2.11.1 getProviderName

Returns the service provider name from the Service descriptor or from the Multilingual service name descriptor using the language preference.

## O.2.12 javax.tv.service.transport.Transport

The object implementing the Transport interface shall also implement the interfaces NetworkCollection and BouquetCollection.

## O.2.13 javax.tv.service.transport.Bouquet

The Bouquet interface is implemented by an object that represents a DVB SI Bouquet.

Objects implementing this interface shall also implement the CAIdentification interface. See O.2.9, "javax.tv.service.navigation.CAIdentification" on page 620.



### O.2.13.1 `getBouquetID`

Returns the integer DVB SI Bouquet ID value.

### O.2.13.2 `getName`

Returns the name of the bouquet from the Bouquet name descriptor or from the Multilingual bouquet name descriptor using the language preference.

### O.2.13.3 `getLocator`

Returns an implementation dependent `javax.tv.locator.Locator` object that does not have a standardized external representation and might not be a `org.davic.net.dvb.DvbLocator`.

## O.2.14 `javax.tv.service.transport.Network`

The Network interface is implemented by an object that represents a DVB SI Network.

### O.2.14.1 `getNetworkID`

Returns the integer DVB SI Network ID value.

### O.2.14.2 `getName`

Returns the name of the network from the Network name descriptor or from the Multilingual network name descriptor using the language preference.

### O.2.14.3 `getLocator`

Returns an implementation dependent `javax.tv.locator.Locator` object that does not have a standardized external representation and might not be a `org.davic.net.dvb.DvbLocator`.

## O.2.15 `javax.tv.service.transport.TransportStream`

The TransportStream interface is implemented by an object that represents a transport stream.

### O.2.15.1 `getTransportStreamID`

Returns the integer DVB SI transport stream ID value.

### O.2.15.2 `getDescription`

Transport streams do not have descriptions in DVB SI, so this method shall return an empty string.

## O.2.16 `javax.tv.service.guide.ProgramEvent`

This interface is implemented by objects representing DVB SI Events.

Objects implementing this interface shall also implement the `CAIdentification` interface. See O.2.9, "[javax.tv.service.navigation.CAIdentification](#)" on page 620.

### O.2.16.1 `getDuration`

Returns the duration value from the event entry in the body of the EIT.

### O.2.16.2 `getStartTime`

Returns the start time value from the event entry in the body of the EIT.

### O.2.16.3 `getEndTime`

Returns the end time value calculated from the start time and duration in the body of the EIT.

#### O.2.16.4 getName

Returns the event name from one of the Short event descriptors in the event using the language preference.

#### O.2.16.5 retrieveDescription

Returns the event description from one of the Short event descriptors in the event using the language preference.

The description text in the `ProgramEventDescription` object is just passed through as a String containing the description as it was transmitted in the EIT table with just a character set mapping performed.

Codes 0x8a and 0xe08a defined in tables A.1 and A.2 of EN 300 468 [4] shall be mapped to the Java newline character, '\n'.

#### O.2.16.6 getRating

Returns the rating from the Parental rating descriptor according to the mapping defined in section O.2.17, "[javax.tv.service.guide.ContentRatingAdvisory](#)" on page 622.

### O.2.17 javax.tv.service.guide.ContentRatingAdvisory

#### O.2.17.1 getDimensionNames

Returns an array containing the string "DVB Age based rating" as one of the elements in the array.

#### O.2.17.2 getRatingLevel

When the parameter is "DVB Age based rating" and the parental rating descriptor contains a rating value between 0x01 and 0x0F for the current region, returns the integer rating value contained in the parental rating descriptor decremented by one (i.e. the value in the descriptor - 1). In other cases when the parameter is "DVB Age based rating" returns -1.

#### O.2.17.3 getRatingText

When the parameter is "DVB Age based rating" and the parental rating descriptor contains a rating value between 0x01 and 0x0F for the current region, returns a string of the form "Recommended minimum age: n years" where n is the rating value in the descriptor incremented by 4 (i.e. the value in the descriptor + 4). In other cases when the parameter is "DVB Age based rating" returns an empty string.

#### O.2.17.4 getDisplayText

When the parental rating descriptor contains a rating value between 0x01 and 0x0F for the current region, returns a string that contains the string "Over n", where n is the rating value in the descriptor incremented by 4 (i.e. the value in the descriptor + 4), as its substring.

## O.3 Integration of the JavaTV SI API and the DVB SI API

In order for the protocol independent service information API to be useful, there needs to be an easy and convenient way for applications to use the DVB specific parts when they are needed. The information provided in the protocol independent API is quite minimal and does not cover all the aspects of the standardized DVB Service Information nor access to the private extensions carried in the standard protocol. If there is no integration between these APIs and the application programmer needs to use a completely different API to retrieve additional information on the object retrieved from the protocol independent API, the usefulness of the protocol independent API is very questionable. In this case, the application programmer will start using only the protocol dependent API, as it provides the complete information and is as easy to use as the other API.

To overcome these problems and make the protocol independent API somehow useful, it needs to be well integrated with the protocol dependent API, so that if an application uses first the protocol independent API for browsing the information, it can easily get additional, protocol dependent information on the objects of interest.

The Java language provides an easy way to achieve this integration: the same objects can implement both the protocol independent interface as well as the protocol dependent interface. This way the application programmer only needs to cast the object to the protocol dependent interface and can directly call methods from the protocol specific API.

Objects implementing the following interfaces of the DVB SI API should implement also the corresponding JavaTV SI API interfaces. When retrieving SI objects through the JavaTV APIs, they shall also implement the corresponding DVB SI API interfaces.

The interfaces of both APIs shall be implemented on the objects as follows:

**Table O.3 :**

org.dvb.si.SINetwork	objects implement also	javax.tv.service.transport.Network
org.dvb.si.SIBouquet	objects implement also	javax.tv.service.transport.Bouquet
org.dvb.si.SITransportStreamNIT	objects implement also	javax.tv.service.transport.TransportStream
org.dvb.si.SIService	objects implement also	javax.tv.service.navigation.ServiceDetails
org.dvb.si.SIEvent	objects implement also	javax.tv.service.guide.ProgramEvent

---

## Annex P (normative): Broadcast Transport Protocol Access

The Object Carousel represents the best suited protocol to carry a structure of objects. Thus, the Object Carousel "mimics" a remote server.

The structure of the objects carried in an Object Carousel is identical to the structure of UU-Objects located on a remote DSMCC-UU Server.

The aim of this API is to enable an application to access files encapsulated in an object carousel or accessible through a DSMCC interactive network. Note that the protocol is abstracted from the application viewpoint, so, objects accessible through this API are either objects encapsulated in an Object Carousel, or Objects located in an interactive DSMCC network on a remote server.

To benefit from the fact that most of the functionalities are already covered by the java.io package, this API inherits from java.io and only defines the extra-functionalities pertaining to:

- a) the nature of the network (broadcast or DSMCC remote server) and its latency (e.g. possibility to asynchronously load the objects)
- b) the type of the objects that can be encapsulated in a carousel and that do not exist in a classical File structure. These are: ServiceGateway, Directory, File, Stream and StreamEvent.
- c) Definition of ServiceGateway, which defines a new namespace corresponding to the new Domain, and enables the mounting of a new volume.

An application can optionally use only the classes of java.io. Alternatively/additionally applications can use additional classes and methods adapted to the specific nature and latency of the network (such as for example, the asynchronous loading of objects).

The following, briefly explains the functionalities offered by this API

The ServiceDomain class enables attaching to a ServiceDomain. Attachment to a serviceDomain corresponds to the mounting of a volume in the file hierarchy system and the loading of the Service Gateway.

When attached to a Service Domain the DSM-CC UU-File, UU-Stream, UU-Directory and UU-StreamEvent objects are accessible through this API.

The class DSMCCObject represents a UU-object. Due to the close relationship between resident files and downloaded files, this class inherits from the java.io.File class. The DSMCCObject class just defines the additional methods specific to DSMCC-UU that basically deal with asynchronous or synchronous loading of Objects.

For the UU-Files or UU-Directory Objects, their content is accessible as it would be for a classical file system, that means through the classical JDK java.io package (e.g., for listing the objects pointed to by a Directory object, you invoke the list() method of the java.io.File class, or to access the content of a UU-File, you can instantiate a FileInputStream to read the File, etc...).

Additionally, the DSMCCStream and DSMCCStreamEvent classes define functionalities specific to the respective types of Objects (Stream and StreamEvent), which basically consists in accessing the attributes of these Objects.

The DSMCCStream class provides access to the following attributes Duration, current NPT. In addition, an application can retrieve the list of Taps (modeled by the 'Locator' class), in order for a Player to be able to control and play that Stream.

The DSMCCStreamEvent class inherits from the DSMCCStream class, and provides access to the event list attributes of a StreamEvent Object. In addition, the application has the possibility to subscribe the events which are present in the eventList.

The AsynchronousLoadingEvent class and its subclasses represent events that are sent to a listener to notify it of the loading of an Object that had been activated by the application (asynchronous loading mode).

The StreamEvent class represents an abstraction of the real event that is generated, i.e. the streameventdescriptor, which enables the broadcaster to synchronize the application with the stream. This class enables the access to the content of an event, the content of the event being described by the StreamEventDescriptor, which is inserted in the stream in DSMCC sections at the transport level.

Finally, the StreamEventListener and AsynchronousLoadingEventListener are interfaces that must be implemented by the application, in order for it to receive the respective StreamEvents and AsynchronousLoadingEvents.

# Package org.dvb.dsmcc

## Description

Provides extended access to files carried in the broadcast stream. It includes some extensions to java.io which are generic to (possibly) long-latency file systems and some concepts which are specific to the DSMCC object carousel.

Class Summary	
<b>Interfaces</b>	
<code>AsynchronousLoadingEventListener</code>	Listener for applications which perform asynchronous loading, in order to be informed if the loading is done or if an error has occurred.
<code>NPTListener</code>	Objects that implement the <code>NPTListener</code> interface can receive <code>NPTEvent</code> events.
<code>ObjectChangeEventListener</code>	The objects that implements the <code>ObjectChangeEventListener</code> interface can receive <code>ObjectChangeEvent</code> event.
<code>StreamEventListener</code>	Objects that implement the <code>StreamEventListener</code> interface can receive <code>StreamEvent</code> event.
<b>Classes</b>	
<code>AsynchronousLoadingEvent</code>	This class described an Object event which is used to notify the loading of a DSMCC object.
<code>DSMCCObject</code>	A <code>DSMCCObject</code> is an object which belongs to a DSMCC ServiceDomain.
<code>DSMCCStream</code>	The <code>Stream</code> class is used to manage DSMCC Stream Objects.
<code>DSMCCStreamEvent</code>	The <code>DSMCCStreamEvent</code> class is used to manage DSMCC StreamEvent Objects.
<code>InvalidFormatEvent</code>	This event is generated if the format of the data received is inconsistent.
<code>InvalidPathnameEvent</code>	The pathname does not exist or the ServiceDomain has been detached.
<code>LoadingAbortedEvent</code>	This event will be sent to the <code>AsynchronousEventListener</code> when an asynchronous loading operation is aborted.
<code>MPEGDeliveryErrorEvent</code>	An <code>MPEGDeliveryErrorEvent</code> indicates that an error (for instance, a time out or accessing the data would require tuning) has occurred while loading data from an MPEG Stream.
<code>NotEntitledEvent</code>	This event is sent when an attempt to asynchronously load an object has failed because the elementary stream carrying the object is scrambled and the user is not entitled to access the content of the object.
<code>NPTDiscontinuityEvent</code>	Sent when an MHP terminal detects a permanent discontinuity in NPT as defined in the main body of this specification.
<code>NPTPresentEvent</code>	Sent when NPT newly appears in a stream where it was not previously present.
<code>NPTRate</code>	Represents the rate at which an NPT time-base progresses.
<code>NPTRateChangeEvent</code>	Sent only when the rate of an NPT time-base changes value.
<code>NPTRemovedEvent</code>	Sent when NPT which was present is removed from a stream.

<b>Class Summary</b>	
<code>NPTStatusEvent</code>	Sent when an MHP terminal detects a change of status in the NPT of a stream.
<code>ObjectChangeEvent</code>	This class describes an object change event that is used to monitor the arrival of a new version of a <code>DSMCCObject</code> .
<code>ServerDeliveryErrorEvent</code>	The local machine can not communicate with the server.
<code>ServiceDomain</code>	A <code>ServiceDomain</code> represents a group of DSMCC objects.
<code>ServiceXFRErrorEvent</code>	The object requested is available in an alternate <code>ServiceDomain</code> .
<code>ServiceXFRReference</code>	A <code>ServiceXFRReference</code> object is used when a DSMCC Object can not be loaded in the current <code>ServiceDomain</code> but is available in an alternate <code>ServiceDomain</code> .
<code>StreamEvent</code>	This class describes a Stream event which is used to synchronize an application with an MPEG Stream.
<code>SuccessEvent</code>	This event indicates that the asynchronous loading was successful.
<b>Exceptions</b>	
<code>DSMCCException</code>	The <code>DSMCCException</code> is the root class of all DSMCC related exceptions
<code>IllegalObjectTypeException</code>	This Exception is thrown when the application attempted to create a <code>DSMCCStream</code> or <code>DSMCCStreamEvent</code> object with an object or a path that did not correspond to a DSMCC Stream or DSMCC StreamEvent respectively
<code>InsufficientResourceException</code>	This exception gets thrown when a request to subscribe to a stream event can not be completed due to resource limitations.
<code>InvalidAddressException</code>	An <code>InvalidAddressException</code> is thrown when the format of an NSAP address is not recognized.
<code>InvalidFormatException</code>	An <code>InvalidFormatException</code> is thrown when an inconsistent DSMCC message is received.
<code>InvalidPathNameException</code>	The <code>InvalidPathNameException</code> is thrown when the path name to a DSMCCObject does not exist or if the <code>ServiceDomain</code> has been detached.
<code>MPEGDeliveryException</code>	An <code>MPEGDeliveryException</code> is thrown when an error (for instance, a time out or accessing the data would require tuning) occurs while loading data from an MPEG Stream.
<code>NotEntitledException</code>	This Exception is thrown when the user is not entitled to access the content of the object (the Elementary Stream is scrambled and the user is not entitled).
<code>NothingToAbortException</code>	A <code>NothingToAbortException</code> is thrown when the abort method is called and there is no loading in progress.
<code>NotLoadedException</code>	A <code>NotLoadedException</code> is thrown when the Stream object constructor is called with a DSMCC Object which is not loaded.
<code>ServerDeliveryException</code>	A <code>ServerDeliveryException</code> is thrown when the local machine can not communicate with the server.
<code>ServiceXFRException</code>	A <code>ServiceXFRException</code> is thrown when a DSMCC Object can not be loaded in the current <code>ServiceDomain</code> but is available in an alternate <code>ServiceDomain</code> (i.e.
<code>UnknownEventException</code>	The <code>UnknownEventException</code> is thrown when a method tries to access to an unknown event.

org.dvb.dsmcc

# AsynchronousLoadingEvent

## Syntax

```
public abstract class AsynchronousLoadingEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.dvb.dsmcc.AsynchronousLoadingEvent
```

## Direct Known Subclasses:

```
InvalidFormatEvent, InvalidPathnameEvent, LoadingAbortedEvent,
MPEGDeliveryErrorEvent, NotEntitledEvent, ServerDeliveryErrorEvent,
ServiceXFRErrorEvent, SuccessEvent
```

## All Implemented Interfaces:

```
java.io.Serializable
```

## Description

This class described an Object event which is used to notify the loading of a DSMCC object.

## Constructors

---

### AsynchronousLoadingEvent(DSMCCObject)

```
public AsynchronousLoadingEvent(DSMCCObject o)
```

Creates an AsynchronousLoadingEvent.

#### Parameters:

- o - the DSMCCObject that generated the event.

## Methods

---

### getSource()

```
public java.lang.Object getSource()
```

Returns the DSMCCObject that generated the event.

#### Overrides:

java.util.EventObject.getSource() in class java.util.EventObject

#### Returns:

the DSMCCObject that generated the event.



org.dvb.dsmcc

# AsynchronousLoadingEventListener

## Syntax

```
public interface AsynchronousLoadingEventListener extends java.util.EventListener
```

## All Superinterfaces:

java.util.EventListener

## Description

Listener for applications which perform asynchronous loading, in order to be informed if the loading is done or if an error has occurred.

## Methods

---

### receiveEvent(AsynchronousLoadingEvent)

```
public void receiveEvent(AsynchronousLoadingEvent e)
```

Method called when an event is sent to the application.

#### Parameters:

e - an AsynchronousLoadingEvent event.

# org.dvb.dsmcc DSMCCException

## Syntax

```
public class DSMCCException extends java.io.IOException
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--java.io.IOException
            |
            +--org.dvb.dsmcc.DSMCCException
```

## Direct Known Subclasses:

`IllegalObjectTypeException`, `InsufficientResourcesException`,  
`InvalidAddressException`, `InvalidFormatException`, `InvalidPathNameException`,  
`MPEGDeliveryException`, `NotEntitledException`, `NothingToAbortException`,  
`NotLoadedException`, `ServerDeliveryException`, `ServiceXFRException`,  
`UnknownEventException`

## All Implemented Interfaces:

`java.io.Serializable`

## Description

The `DSMCCException` is the root class of all DSMCC related exceptions

## Constructors

---

### DSMCCException()

```
public DSMCCException()
```

Construct a `DSMCCException` with no detail message

---

### DSMCCException(String)

```
public DSMCCException(java.lang.String s)
```

Construct a `DSMCCException` with the specified detail message

#### Parameters:

`s` - the detail message

# org.dvb.dsmcc DSMCCObject

## Syntax

```
public class DSMCCObject extends java.io.File
```

```
java.lang.Object
|
+--java.io.File
|
+--org.dvb.dsmcc.DSMCCObject
```

## All Implemented Interfaces:

java.io.Serializable

## Description

A DSMCCObject is an object which belongs to a DSMCC ServiceDomain. As soon as a ServiceDomain has been attached to the file system hierarchy, DSMCCObject objects can be created to access the ServiceDomain objects. Paths used to create a DSMCCObject can be either absolute or relative as defined in detail in the main body of this specification. To access the content of the object:

- For a Directory, the method list of the java.io.File class has to be used to get the entries of the directory.
- For a Stream object, the class DSMCCStream has to be used.
- For a File, the java.io.FileInputStream class or the java.io.RandomAccessFile has to be used.

NB:

- Obviously, for the Object Carousel, the write mode of java.io.RandomAccessFile is not allowed.

## See Also:

[ServiceDomain](#)

## Fields

---

### FROM\_CACHE

```
public static final int FROM_CACHE
```

Constant to indicate that the data for an object shall only be retrieved where it is already in cache and meets the requirements of cache priority signaling. Where data is not in the cache, or the contents don't meet the requirements of the of cache priority signaling, attempts to load a DSMCCObject shall fail.

**Since:**

MHP 1.0.1

---

### FROM\_CACHE\_OR\_STREAM

```
public static final int FROM_CACHE_OR_STREAM
```

Constant to indicate that the data for an object shall be automatically be retrieved from the network where the data is not already cached. Note that this method does not modify the caching policy controlled by the signaling in the OC. So, if the data is signalled as requiring transparent caching then data will be retrieved from the network if required.

**Since:**

MHP 1.0.1

---

**FROM\_STREAM\_ONLY**

```
public static final int FROM_STREAM_ONLY
```

Constant to indicate that the data for an object shall always be retrieved from the network even if the data has already been cached.

**Since:**

MHP 1.0.1

## Constructors

---

**DSMCCObject(DSMCCObject, String)**

```
public DSMCCObject(DSMCCObject dir, java.lang.String name)
```

Create a DSMCCObject object.

**Parameters:**

`dir` - the directory object.

`name` - the filename.

---

**DSMCCObject(String)**

```
public DSMCCObject(java.lang.String path)
```

Create a DSMCCObject object.

**Parameters:**

`path` - the path to the file.

---

**DSMCCObject(String, String)**

```
public DSMCCObject(java.lang.String path, java.lang.String name)
```

Create a DSMCCObject object.

**Parameters:**

`path` - the directory Path.

`name` - the filename.

## Methods

---

**abort()**

```
public void abort()
```

This method is used to abort a load in progress. It can be used to abort either a synchronousLoad or an asynchronousLoad.

**Throws:**

`NothingToAbortException` - There is no loading in progress.

---

### **addObjectChangeListener(ObjectChangeListener)**

```
public void addObjectChangeListener(ObjectChangeListener listener)
```

Subscribes an `ObjectChangeListener` to receive notifications of version changes of `DSMCCObject`.

This listener shall never be fired until after the object has successfully entered the loaded state for the first time. Hence objects which never successfully enter the loaded state (e.g. because the object cannot be found) shall never have this listener fire. Once an object has successfully entered the loaded state once, this event shall continue to be fired when changes are detected by the MHP regardless of further transitions in or out of the loaded state.

**Parameters:**

`listener` - the `ObjectChangeListener` to be notified .

---

### **asynchronousLoad(AsynchronousLoadingEventListener)**

```
public void asynchronousLoad(AsynchronousLoadingEventListener l)
```

This method is used to asynchronously load a carousel object. For each call to this method which returns without throwing an exception, one of the following events will be sent to the application (by a listener mechanism) as soon as the loading is done or if an error has occurred: `SuccessEvent`, `InvalidFormatEvent`, `InvalidPathNameEvent`, `MPEGDeliveryErrorEvent`, `ServerDeliveryErrorEvent`, `ServiceXFRErrorEvent`, `NotEntitledEvent`

**Parameters:**

`l` - an `AsynchronousLoadingEventListener` to receive events related to asynchronous loading.

**Throws:**

`InvalidPathNameException` - the object can not be found.

---

### **getSigners()**

```
public java.security.cert.X509Certificate[][] getSigners()
```

This method will return the lists of certificate chains that can authenticate the `DSMCCObject`. If the `DSMCCObject` is not loaded, this method will return null. If the `DSMCCObject` is loaded but not authenticated this method will return an outer array of size zero.

**Returns:**

a two-dimensional array of X.509 certificates, where the first index of the array determines a certificate chain and the second index identifies the certificate within the chain. Within one certificate chain the leaf certificate is first followed by any intermediate certificate authorities in the order of the chain with the root CA certificate as the last item.

**Since:**

MHP 1.0.1

---

### **getURL()**

```
public java.net.URL getURL()
```

Returns a URL identifying this carousel object. If the directory entry for the object has not been loaded then null shall be returned.

**Returns:**

a URL identifying the carousel object or null

**Since:**

MHP 1.0.1

---

**isLoading()**

```
public boolean isLoading()
```

Returns a boolean indicating whether or not the DSMCCObject has been loaded.

**Returns:**

true if the file is already loaded, false otherwise.

---

**isObjectKindKnown()**

```
public boolean isObjectKindKnown()
```

Returns a boolean indicating if the kind of the object is known. (The kind of an object is known if the directory containing it is loaded).

**Returns:**

true if the type of the object is known, false otherwise.

---

**isStream()**

```
public boolean isStream()
```

Returns a boolean indicating whether or not the DSMCCObject is a DSMCC Stream object.

**Returns:**

true if the file is a stream, false if the object is not a stream or if the object kind is unknown.

---

**isStreamEvent()**

```
public boolean isStreamEvent()
```

Returns a boolean indicating whether or not the DSMCCObject is a DSMCC StreamEvent object.  
NB: If isStreamEvent is true then isStream is true also.

**Returns:**

true if the file is a stream event, false if the object is not a stream event or if the object kind is unknown.

---

**loadDirectoryEntry(AsynchronousLoadingEventListener)**

```
public void loadDirectoryEntry(AsynchronousLoadingEventListener l)
```

Asynchronous loading of the directory entry information. Calling this is equivalent of calling the method asynchronousLoad on the parent directory of a DSMCCObject.

**Parameters:**

l - a listener which will be called when the loading is done.

**Throws:**

`InvalidPathNameException` - if the object cannot be found.

---

**prefetch(DSMCCObject, String, byte)**

```
public static boolean prefetch(DSMCCObject dir, java.lang.String path, byte priority)
```

Calling this method will issue a hint to the MHP for pre-fetching the object data for that DSMCC object into cache.

**Parameters:**

`dir` - the directory object in which to pre-fetch the data.

`path` - the relative path name of object to pre-fetch, starting from the directory object passes as parameter.

`priority` - the relative priority of this pre-fetch request (higher = more important)

**Returns:**

true if the MHP supports pre-fetching (i.e. will try to process the request) and false otherwise. Note that a return value of 'true' is only an indication that the MHP receiver supports pre-fetching. It is not a guarantee that the requested data will actually be loaded into cache as the receiver may decide to drop the request in order to make resources available for regular load requests.

**prefetch(String, byte)**

```
public static boolean prefetch(java.lang.String path, byte priority)
```

Calling this method will issue a hint to the MHP for pre-fetching the object data for that DSMCC object into cache.

**Parameters:**

`path` - the absolute pathname of the object to pre-fetch.

`priority` - the relative priority of this pre-fetch request (higher = more important)

**Returns:**

true if the MHP supports pre-fetching (i.e. will try to process the request) and false otherwise. Note that a return value of 'true' is only an indication that the MHP receiver supports pre-fetching. It is not a guarantee that the requested data will actually be loaded into cache as the receiver may decide to drop the request in order to make resources available for regular load requests.

**removeObjectChangeListener(ObjectChangeListener)**

```
public void removeObjectChangeListener(ObjectChangeListener listener)
```

Unsubscribes an ObjectChangeListener to receive notifications of version changes of DSMCCObject.

**Parameters:**

`listener` - a previously registered ObjectChangeListener.

**setRetrievalMode(int)**

```
public void setRetrievalMode(int retrieval_mode)
```

Set the retrieval mode for a DSMCCObject. The default retrieval mode is FROM\_CACHE\_OR\_STREAM. The retrieval mode state is sampled when the object is loaded (whether explicitly or as described in "Constraints on the java.io.File methods for broadcast carousels"). Changing the retrieval mode for a loaded object has no effect until the object is unloaded and loaded again.

**Parameters:**

`retrieval_mode` - the retrieval mode to be used for the object specified as one of the public static final constants in this class.

**Throws:**

`IllegalArgumentException` - if the `retrieval_mode` specified is not one listed defined for use with this method.

**Since:**

MHP 1.0.1

---

**synchronousLoad()**

```
public void synchronousLoad()
```

This method is used to load a DSMCCObject. This method blocks until the file is loaded. It can be aborted from another thread with the abort method. In this case the InterruptedIOException is thrown. If the IOR of the object itself or one of its parent directories is a Lite Option Profile Body, the MHP implementation will not attempt to resolve it: a ServiceXFRException is thrown to indicate to the application where the DSMCCObject is actually located.

**Throws:**

`InterruptedIOException` - the loading has been aborted.

`InvalidPathNameException` - the Object can not be found.

`NotEntitledException` - the stream carrying the object is scrambled and the user has no entitlements to descramble the stream.

`ServiceXFRException` - the IOR of the object or one of its parent directories is a Lite Option Profile Body.

`InvalidFormatException` - an inconsistent DSMCC message has been received.

`MPEGDeliveryException` - an error has occurred while loading data from MPEG stream such as a timeout

`ServerDeliveryException` - when an MHP terminal cannot communicate with the server for files delivered over a bi-directional IP connection.

---

**unload()**

```
public void unload()
```

When calling this method, the applications gives a hint to the MHP that if this object is not consumed by another application/thread, the system can free all the resources allocated to this object. It is worth noting that if other clients use this object (e.g. a file input stream is opened on this object or if the corresponding stream or stream event is being consumed) the system resources allocated to this object will not be freed.

**Throws:**

`NotLoadedException` - the carousel object is not loaded.



# org.dvb.dsmcc DSMCCStream

## Syntax

```
public class DSMCCStream
|
| java.lang.Object
| |
| | |--org.dvb.dsmcc.DSMCCStream
```

## Direct Known Subclasses:

[DSMCCStreamEvent](#)

## Description

The Stream class is used to manage DSMCC Stream Objects.

## See Also:

[DSMCCObject](#)

## Constructors

---

### DSMCCStream(DSMCCObject)

```
public DSMCCStream(DSMCCObject aDSMCCObject)
```

Creates a Stream Object from a DSMCC Object. The BIOP message referenced by the DSMCCObject has to be a Stream or StreamEvent BIOP message.

#### Parameters:

`aDSMCCObject` - the DSMCC object which describes the stream

#### Throws:

[NotLoadedException](#) - the DSMCCObject is not loaded.

[IllegalObjectTypeException](#) - the DSMCCObject is neither a DSMCC Stream nor a DSMCCStreamEvent

---

### DSMCCStream(String)

```
public DSMCCStream(java.lang.String path)
```

Create a Stream Object from its pathname. For an object Carousel, this method will block until the module which contains the object is loaded. The BIOP message referenced by the DSMCCObject pointed to by the parameter `path` has to be a Stream or StreamEvent BIOP message. The path has to lead to a DSMCCStream (or a DSMCCStreamEvent)

#### Parameters:

`path` - the pathname of the DSMCCStream Object.

#### Throws:

[IOException](#) - If an IO error occurred.

[IllegalObjectTypeException](#) - the DSMCCObject is neither a DSMCC Stream nor a DSMCCStreamEvent

---

**DSMCCStream(String, String)**

```
public DSMCCStream(java.lang.String path, java.lang.String name)
```

Create a DSMCCStream from its pathname. For an object Carousel, this method will block until the module which contains the object is loaded. The BIOP message referenced by the DSMCCObject pointed to be the parameters path and name has to be a Stream or StreamEvent BIOP message. The path has to lead to a DSMCCStream (or a DSMCCStreamEvent)

**Parameters:**

path - the directory path.

name - the name of the DSMCCStream Object.

**Throws:**

`IOException` - If an IO error occurred.

`IllegalObjectTypeException` - the DSMCCObject is neither a DSMCC Stream nor a DSMCCStreamEvent

---

**Methods**

---

**addNPTListener(NPTListener)**

```
public void addNPTListener(NPTListener l)
```

Add a listener to NPT events on the `DSMCCStream` object. Adding the same listener a second time has no effect.

**Parameters:**

l - the listener

**Since:**

MHP 1.0.1

---

**getDuration()**

```
public long getDuration()
```

This function returns the duration in milliseconds of the DSMCC Stream. If the DSMCCStream BIOP message doesn't specify duration, zero will be returned.

**Returns:**

The duration in milliseconds of the DSMCC Stream.

---

**getNPT()**

```
public long getNPT()
```

This function is used to get the current NPT in milliseconds.

**Returns:**

the current NPT in milliseconds or zero if DSMCC Stream object BIOP message doesn't contain any taps pointing to NPT reference descriptors.

**Throws:**

`MPEGDeliveryException` - if there's an error in retrieving NPT reference descriptors

---

**getNPTRate()**

```
public NPTRate getNPTRate()
```

Get the NPT rate for the `DSMCCStream` object. Returns null if the DSMCC stream has no associated NPT rate (i.e. no `STR_NPT_USE` tap in the list of taps).

**Returns:**

the NPT rate or null

**Since:**

MHP 1.0.1

---

**getStreamLocator()**

```
public org.davic.net.Locator getStreamLocator()
```

This function returned a `Locator` referencing the streams of this collection. The interpretation of the return value is determined by the `isMPEGProgram` method.

**Returns:**

a locator.

---

**isAudio()**

```
public boolean isAudio()
```

This function returns a boolean indicating if the Stream Object refers to an audio stream. This is the case if the `audio` field in the `Stream(Event)` BIOP message has a value different from zero.

**Returns:**

true only if the Stream object refers to an audio stream

---

**isData()**

```
public boolean isData()
```

This function returns a boolean indicating if the Stream Object refers to a data stream. This is the case if the `data` field in the `Stream(Event)` BIOP message has a value different from zero.

**Returns:**

true only if the Stream object refers to a data stream

---

**isMPEGProgram()**

```
public boolean isMPEGProgram()
```

This method will return true if the `Stream(Event)` BIOP message contains a tap with use field `BIOP_PROGRAM_USE`, otherwise it will return false.

**Returns:**

true only if the `Stream(Event)` BIOP message is as described above

---

**isVideo()**

```
public boolean isVideo()
```

This function returns a boolean indicating if the Stream Object refers to an video stream. This is the case if the `video` field in the `Stream(Event)` BIOP message has a value different from zero otherwise false is returned.

**Returns:**

true only if the Stream object refers to an video stream

---

**removeNPTListener(NPTListener)**

```
public void removeNPTListener(NPTListener l)
```

Remove a listener to NPT events on the `DSMCCStream` object. Removing a non-subscribed listener has no effect.

**Parameters:**

l - the listener to remove

**Since:**

MHP 1.0.1

# org.dvb.dsmcc DSMCCStreamEvent

## Syntax

```
public class DSMCCStreamEvent extends DSMCCStream
```

```
java.lang.Object
|
+--DSMCCStream
|
+--org.dvb.dsmcc.DSMCCStreamEvent
```

## Description

The DSMCCStreamEvent class is used to manage DSMCC StreamEvent Objects. Applications wishing to monitor changes in the list of events which are part of this stream event should use `DSMCCObject.addObjectChangeListener` on the `DSMCCObject` representing which describes this set of stream events.

## Constructors

---

### DSMCCStreamEvent(DSMCCObject)

```
public DSMCCStreamEvent(DSMCCObject aDSMCCObject)
```

Create a `DSMCCStreamEvent` from a `DSMCCObject`. The Object has to be a `DSMCCStreamEvent`.

#### Parameters:

`aDSMCCObject` - the DSMCC object which describes the stream.

#### Throws:

`NotLoadedException` - the `DSMCCObject` is not loaded.

`IllegalObjectTypeException` - the `DSMCCObject` does not lead to a `DSMCCStreamEvent`.

---

### DSMCCStreamEvent(String)

```
public DSMCCStreamEvent(java.lang.String path)
```

Create a `DSMCCStreamEvent` Object from its pathname. The path has to lead to a `DSMCCStreamEvent`.

#### Parameters:

`path` - the pathname of the `DSMCCStreamEvent` object

#### Throws:

`IOException` - An IO error has occurred.

`IllegalObjectTypeException` - the path does not lead to a `DSMCCStreamEvent`.

---

### DSMCCStreamEvent(String, String)

```
public DSMCCStreamEvent(java.lang.String path, java.lang.String name)
```

Create a DSMCCStreamEvent from its pathname. For an object Carousel, this method will block until the module which contains the object is loaded. The path has to lead to a DSMCC Stream Event

**Parameters:**

path - the directory path.

name - the name of the DSMCCStreamEvent Object.

**Throws:**

IOException - If an IO error occurred.

IllegalObjectTypeException - the path does not lead to a DSMCC StreamEvent.

## Methods

---

**getEventList()**

```
public java.lang.String[] getEventList()
```

This function is used to get the list of the events of the DSMCCStreamEvent object.

**Returns:**

The list of the eventName.

---

**subscribe(String, StreamEventListener)**

```
public synchronized int subscribe(java.lang.String eventName, StreamEventListener l)
```

This function is used to subscribe to an event of a DSMCC StreamEvent object.

**Parameters:**

eventName - the name of the event.

l - an object that implements the StreamEventListener Interface.

**Returns:**

The event Identifier.

**Throws:**

UnknownEventException - the event cannot be found at this time

InsufficientResourcesException - if resources needed to perform the subscription are not available

---

**unsubscribe(int, StreamEventListener)**

```
public synchronized void unsubscribe(int eventId, StreamEventListener l)
```

This function is used to cancel the subscription to an event of a DSMCCEvent object.

**Parameters:**

eventId - Identifier of the event.

l - an object that implements the StreamEventListener Interface.

**Throws:**

UnknownEventException - The event can not be found.

---

**unsubscribe(String, StreamEventListener)**

```
public synchronized void unsubscribe(java.lang.String eventName, StreamEventListener l)
```

This function is used to cancel the subscription to an event of a DSMCCEvent object.

**Parameters:**

eventName - the name of the event.

l - an object that implements the StreamEventListener Interface.

**Throws:**

`UnknownEventException` - The event can not be found.

org.dvb.dsmcc

# IllegalObjectTypeException

## Syntax

```
public class IllegalObjectTypeException extends DSMCCException
```

```

java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--java.io.IOException
            |
            +--DSMCCException
                |
                +--org.dvb.dsmcc.IllegalObjectTypeException

```

## All Implemented Interfaces:

```
java.io.Serializable
```

## Description

This Exception is thrown when the application attempted to create a DSMCCStream or DSMCCStreamEvent object with an object or a path that did not correspond to a DSMCC Stream or DSMCC StreamEvent respectively

## Constructors

---

### IllegalObjectTypeException()

```
public IllegalObjectTypeException()
```

constructor of the exception with no detail message

---

### IllegalObjectTypeException(String)

```
public IllegalObjectTypeException(java.lang.String s)
```

constructor of the exception

#### Parameters:

s - detail message



org.dvb.dsmcc

# InsufficientResourcesException

## Syntax

```
public class InsufficientResourcesException extends DSMCCException
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--java.io.IOException
            |
            +--DSMCCException
                |
                +--org.dvb.dsmcc.InsufficientResourcesException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This exception gets thrown when a request to subscribe to a stream event cannot be completed due to resource limitations. For example, no section filters or system timers may be available. This exception will not get thrown when there is not enough memory available to complete the request - this will get signalled by a `java.lang.OutOfMemoryError`

## Since:

MHP 1.0.1

## Constructors

---

### InsufficientResourcesException()

```
public InsufficientResourcesException()
```

Construct an `InsufficientResourcesException` with no detail message

---

### InsufficientResourcesException(String)

```
public InsufficientResourcesException(java.lang.String message)
```

Construct an `InsufficientResourcesException` with the specified detail message

#### Parameters:

`message` - the message for the exception

# org.dvb.dsmcc InvalidAddressException

## Syntax

```
public class InvalidAddressException extends DSMCCException
```

```
java.lang.Object
|
+--java.lang.Throwable
   |
   +--java.lang.Exception
      |
      +--java.io.IOException
         |
         +--DSMCCException
            |
            +--org.dvb.dsmcc.InvalidAddressException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

An InvalidAddressException is thrown when the format of an NSAP address is not recognized.

## Constructors

---

### InvalidAddressException()

```
public InvalidAddressException()
```

Construct a InvalidAddressException with no detail message

---

### InvalidAddressException(String)

```
public InvalidAddressException(java.lang.String s)
```

Construct a InvalidAddressException with the specified detail message

#### Parameters:

*s* - the detail message

# org.dvb.dsmcc InvalidFormatEvent

## Syntax

```
public class InvalidFormatEvent extends AsynchronousLoadingEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--AsynchronousLoadingEvent
|
+--org.dvb.dsmcc.InvalidFormatEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This event is generated if the format of the data received is inconsistent.

## Constructors

---

### InvalidFormatEvent(DSMCCObject)

```
public InvalidFormatEvent(DSMCCObject o)
```

Create an InvalidFormatException object.

#### Parameters:

- o - the DSMCCObject that generated the event.

## Methods

---

### getSource()

```
public java.lang.Object getSource()
```

Returns the DSMCCObject that generated the event

#### Overrides:

`getSource()` in class `AsynchronousLoadingEvent`

#### Returns:

the DSMCCObject that generated the event

# org.dvb.dsmcc InvalidFormatException

## Syntax

```
public class InvalidFormatException extends DSMCCException
```

```
java.lang.Object
|
+--java.lang.Throwable
   |
   +--java.lang.Exception
      |
      +--java.io.IOException
         |
         +--DSMCCException
            |
            +--org.dvb.dsmcc.InvalidFormatException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

An InvalidFormatException is thrown when an inconsistent DSMCC message is received.

## Constructors

---

### InvalidFormatException()

```
public InvalidFormatException()
```

Construct an InvalidFormatException with no detail message

---

### InvalidFormatException(String)

```
public InvalidFormatException(java.lang.String s)
```

Construct an InvalidFormatException with the specified detail message

#### Parameters:

*s* - the detail message

# org.dvb.dsmcc InvalidPathnameEvent

## Syntax

```
public class InvalidPathnameEvent extends AsynchronousLoadingEvent
```

```
java.lang.Object
|
+--java.util.EventObject
    |
    +--AsynchronousLoadingEvent
        |
        +--org.dvb.dsmcc.InvalidPathnameEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

The pathname does not exist or the ServiceDomain has been detached.

## Constructors

---

### InvalidPathnameEvent(DSMCCObject)

```
public InvalidPathnameEvent(DSMCCObject o)
```

Create an InvalidPathnameEvent.

#### Parameters:

- o - the DSMCCObject that generated this event.

## Methods

---

### getSource()

```
public java.lang.Object getSource()
```

Returns the DSMCCObject that generated the event.

#### Overrides:

`getSource()` in class `AsynchronousLoadingEvent`

#### Returns:

the DSMCCObject that generated the event.

# org.dvb.dsmcc InvalidPathNameException

## Syntax

```
public class InvalidPathNameException extends DSMCCException
```

```
java.lang.Object
|
+--java.lang.Throwable
   |
   +--java.lang.Exception
      |
      +--java.io.IOException
         |
         +--DSMCCException
            |
            +--org.dvb.dsmcc.InvalidPathNameException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

The InvalidPathNameException is thrown when the path name to a DSMCCObject does not exist or if the ServiceDomain has been detached.

## Constructors

---

### InvalidPathNameException()

```
public InvalidPathNameException()
```

Construct an InvalidPathNameException with no detail message

---

### InvalidPathNameException(String)

```
public InvalidPathNameException(java.lang.String s)
```

Construct an InvalidPathNameException with the specified detail message

#### Parameters:

s - the detail message

# org.dvb.dsmcc LoadingAbortedEvent

## Syntax

```
public class LoadingAbortedEvent extends AsynchronousLoadingEvent
```

```
java.lang.Object
|
+--java.util.EventObject
    |
    +--AsynchronousLoadingEvent
        |
        +--org.dvb.dsmcc.LoadingAbortedEvent
```

## All Implemented Interfaces:

```
java.io.Serializable
```

## Description

This event will be sent to the `AsynchronousEventListener` when an asynchronous loading operation is aborted.

## Since:

MHP 1.0.1

## Constructors

---

### LoadingAbortedEvent(DSMCCObject)

```
public LoadingAbortedEvent(DSMCCObject aDSMCCObject)
```

Creates a `LoadingAbortedEvent` object.

#### Parameters:

`aDSMCCObject` - the `DSMCCObject` that generated the event.

## Methods

---

### getSource()

```
public java.lang.Object getSource()
```

Returns the `DSMCCObject` that generated the event.

#### Overrides:

`getSource()` in class `AsynchronousLoadingEvent`

#### Returns:

the `DSMCCObject` whose loading was aborted

# org.dvb.dsmcc MPEGDeliveryErrorEvent

## Syntax

```
public class MPEGDeliveryErrorEvent extends AsynchronousLoadingEvent
```

```
java.lang.Object
|
+--java.util.EventObject
    |
    +--AsynchronousLoadingEvent
        |
        +--org.dvb.dsmcc.MPEGDeliveryErrorEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

An MPEGDeliveryErrorEvent indicates that an error (for instance, a time out or accessing the data would require tuning) has occurred while loading data from an MPEG Stream.

## Constructors

---

### MPEGDeliveryErrorEvent(DSMCCObject)

```
public MPEGDeliveryErrorEvent(DSMCCObject o)
```

Creates an MPEGDeliveryEvent.

#### Parameters:

- o - the DSMCCObject that generated the event.

## Methods

---

### getSource()

```
public java.lang.Object getSource()
```

Returns the DSMCCObject that generated the event.

#### Overrides:

getSource() in class AsynchronousLoadingEvent

#### Returns:

the DSMCCObject that generated the event.



# org.dvb.dsmcc MPEGDeliveryException

## Syntax

```
public class MPEGDeliveryException extends DSMCCException
```

```
java.lang.Object
|
+--java.lang.Throwable
|
+--java.lang.Exception
|
+--java.io.IOException
|
+--DSMCCException
|
+--org.dvb.dsmcc.MPEGDeliveryException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

An MPEGDeliveryException is thrown when an error (for instance, a time out or accessing the data would require tuning) occurs while loading data from an MPEG Stream.

## Constructors

---

### MPEGDeliveryException()

```
public MPEGDeliveryException()
```

Construct an MPEGDeliveryException with no detail message

---

### MPEGDeliveryException(String)

```
public MPEGDeliveryException(java.lang.String s)
```

Construct an MPEGDeliveryException with the specified detail message

#### Parameters:

s - the detail message

# org.dvb.dsmcc NotEntitledEvent

## Syntax

```
public class NotEntitledEvent extends AsynchronousLoadingEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--AsynchronousLoadingEvent
|
+--org.dvb.dsmcc.NotEntitledEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This event is sent when an attempt to asynchronously load an object has failed because the elementary stream carrying the object is scrambled and the user is not entitled to access the content of the object.

## Constructors

---

### NotEntitledEvent(DSMCCObject)

```
public NotEntitledEvent(DSMCCObject o)
```

Creates a NotEntitledEvent object.

#### Parameters:

- o - the DSMCCObject that generated the event.

## Methods

---

### getSource()

```
public java.lang.Object getSource()
```

Returns the DSMCCObject that generated the event.

#### Overrides:

`getSource()` in class `AsynchronousLoadingEvent`

#### Returns:

the DSMCCObject that generated the event.

# org.dvb.dsmcc NotEntitledException

## Syntax

```
public class NotEntitledException extends DSMCCException
```

```
java.lang.Object
|
+--java.lang.Throwable
   |
   +--java.lang.Exception
      |
      +--java.io.IOException
         |
         +--DSMCCException
            |
            +--org.dvb.dsmcc.NotEntitledException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This Exception is thrown when the user is not entitled to access the content of the object (the Elementary Stream is scrambled and the user is not entitled).

## Constructors

---

### NotEntitledException()

```
public NotEntitledException()
```

construct a NotEntitledException with no detail message

---

### NotEntitledException(String)

```
public NotEntitledException(java.lang.String s)
```

construct a NotEntitledException with a detail message

#### Parameters:

s - detail message

# org.dvb.dsmcc NothingToAbortException

## Syntax

```
public class NothingToAbortException extends DSMCCException
```

```
java.lang.Object
|
+--java.lang.Throwable
   |
   +--java.lang.Exception
      |
      +--java.io.IOException
         |
         +--DSMCCException
            |
            +--org.dvb.dsmcc.NothingToAbortException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

A NothingToAbortException is thrown when the abort method is called and there is no loading in progress.

## Constructors

---

### NothingToAbortException()

```
public NothingToAbortException()
```

Construct a NothingToAbortException with no detail message

---

### NothingToAbortException(String)

```
public NothingToAbortException(java.lang.String s)
```

Construct a NothingToAbortException with the specified detail message

#### Parameters:

s - the detail message

# org.dvb.dsmcc NotLoadedException

## Syntax

```
public class NotLoadedException extends DSMCCException
```

```
java.lang.Object
|
+--java.lang.Throwable
   |
   +--java.lang.Exception
      |
      +--java.io.IOException
         |
         +--DSMCCException
            |
            +--org.dvb.dsmcc.NotLoadedException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

A NotLoadedException is thrown when the Stream object constructor is called with a DSMCC Object which is not loaded.

## Constructors

---

### NotLoadedException()

```
public NotLoadedException()
```

Construct a NotLoadedException with no detail message

---

### NotLoadedException(String)

```
public NotLoadedException(java.lang.String s)
```

Construct a NotLoadedException with the specified detail message

#### Parameters:

s - the detail message

# org.dvb.dsmcc NPTDiscontinuityEvent

## Syntax

```
public class NPTDiscontinuityEvent extends NPTStatusEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--NPTStatusEvent
|
+--org.dvb.dsmcc.NPTDiscontinuityEvent
```

## All Implemented Interfaces:

```
java.io.Serializable
```

## Description

Sent when an MHP terminal detects a permanent discontinuity in NPT as defined in the main body of this specification. This represents an error condition in the incoming broadcast.

This event shall be sent following a PCR discontinuity when the MHP terminal has enough information to determine that there will be an NPT discontinuity. If this is because of invalid data in a new NPTReferenceDescriptor then the event will be generated when that new NPTReferenceDescriptor is detected by the MHP terminal. If this is because no new NPTReferenceDescriptor is detected within the time allowed by the main body of this specification then it will be generated when that time interval has elapsed.

## Since:

```
MHP 1.0.1
```

## Constructors

---

### NPTDiscontinuityEvent(DSMCCStream, long, long)

```
public NPTDiscontinuityEvent(DSMCCStream source, long before, long after)
```

Construct an event. The *before* and *after* values used shall be the values at the time when the receiver determined that a NPT discontinuity has happened. If this is because of invalid data in a new NPTReferenceDescriptor then this is the time when that new descriptor was known to be invalid. If this is because of the absence of a new NPTReferenceDescriptor then this will be when the MHP terminal detects that the time interval allowed by this specification for such new descriptors has elapsed

#### Parameters:

- source* - the stream whose NPT suffered a discontinuity
- before* - the last NPT value detected before the discontinuity
- after* - the first NPT value detected after the discontinuity

## Methods

---

### **getFirstNPT()**

```
public long getFirstNPT()
```

Return the first known stable value of NPT after the discontinuity

**Returns:**

an NPT value

---

### **getLastNPT()**

```
public long getLastNPT()
```

Return the last known stable value of NPT before the discontinuity

**Returns:**

an NPT value

# org.dvb.dsmcc NPTListener

## Syntax

```
public interface NPTListener extends java.util.EventListener
```

## All Superinterfaces:

```
java.util.EventListener
```

## Description

Objects that implement the `NPTListener` interface can receive `NPTEvent` events.

## Since:

MHP 1.0.1

## Methods

---

### **receiveNPTStatusEvent(NPTStatusEvent)**

```
public void receiveNPTStatusEvent(NPTStatusEvent e)
```

Send a `NPTStatusEvent` to a registered listener.

#### **Parameters:**

`e` - a `NPTStatusEvent` describing the status change

---

### **receiveRateChangedEvent(NPTRateChangeEvent)**

```
public void receiveRateChangedEvent(NPTRateChangeEvent e)
```

Send a `NPTRateChangeEvent` to a registered listener.

#### **Parameters:**

`e` - the `NPTRateChangeEvent` event.



# org.dvb.dsmcc NPTPresentEvent

## Syntax

```
public class NPTPresentEvent extends NPTStatusEvent
```

```
java.lang.Object  
|  
+--java.util.EventObject  
|  
+--NPTStatusEvent  
|  
+--org.dvb.dsmcc.NPTPresentEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Sent when NPT newly appears in a stream where it was not previously present.

## Since:

MHP 1.0.1

## Constructors

---

### NPTPresentEvent(DSMCCStream)

```
public NPTPresentEvent(DSMCCStream source)
```

Construct an event.

#### Parameters:

source - the stream where NPT appeared

# org.dvb.dsmcc NPTRate

## Syntax

```
public class NPTRate
```

```
java.lang.Object
```

```
|
```

```
+--org.dvb.dsmcc.NPTRate
```

## Description

Represents the rate at which an NPT time-base progresses. Rates are expressed as the combination of a numerator and a denominator. Instances of this class are constructed by the platform and returned to applications.

## Since:

MHP 1.0.1

## Methods

---

### getDenominator()

```
public int getDenominator()
```

Get the NPT rate's denominator.

#### Returns:

the denominator

---

### getNumerator()

```
public int getNumerator()
```

Get the NPT rate's numerator. A value of zero indicates that the NPT is not progressing.

#### Returns:

the numerator

# org.dvb.dsmcc NPTRateChangeEvent

## Syntax

```
public class NPTRateChangeEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.dvb.dsmcc.NPTRateChangeEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Sent only when the rate of an NPT time-base changes value.

## Since:

MHP 1.0.1

## Constructors

---

### NPTRateChangeEvent(DSMCCStream, NPTRate)

```
public NPTRateChangeEvent(DSMCCStream source, NPTRate rate)
```

Construct an event.

#### Parameters:

`source` - the stream whose rate changed

`rate` - the new rate of that stream immediately following the change

## Methods

---

### getRate()

```
public NPTRate getRate()
```

Return the new rate of the stream immediately after the change.

#### Returns:

a `NPTRate` object encapsulating the new rate

---

### getSource()

```
public java.lang.Object getSource()
```

Return the stream whose rate changed.

#### Overrides:

`java.util.EventObject.getSource()` in class `java.util.EventObject`

#### Returns:

the `DSMCCStream` object on which the rate change has occurred.

# org.dvb.dsmcc NPTRemovedEvent

## Syntax

```
public class NPTRemovedEvent extends NPTStatusEvent
```

```
java.lang.Object  
|  
+--java.util.EventObject  
|  
+--NPTStatusEvent  
|  
+--org.dvb.dsmcc.NPTRemovedEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Sent when NPT which was present is removed from a stream.

## Since:

MHP 1.0.1

## Constructors

---

### NPTRemovedEvent(DSMCCStream)

```
public NPTRemovedEvent(DSMCCStream source)
```

Construct an event.

#### Parameters:

`source` - the stream which NPT was removed from

# org.dvb.dsmcc NPTStatusEvent

## Syntax

```
public abstract class NPTStatusEvent extends java.util.EventObject
```

```
java.lang.Object  
|  
+--java.util.EventObject  
|  
+--org.dvb.dsmcc.NPTStatusEvent
```

## Direct Known Subclasses:

[NPTDiscontinuityEvent](#), [NPTPresentEvent](#), [NPTRemovedEvent](#)

## All Implemented Interfaces:

[java.io.Serializable](#)

## Description

Sent when an MHP terminal detects a change of status in the NPT of a stream.

## Since:

MHP 1.0.1

## Constructors

---

### NPTStatusEvent(DSMCCStream)

```
public NPTStatusEvent(DSMCCStream source)
```

Construct an event.

#### Parameters:

`source` - the stream whose NPT status changed

## Methods

---

### getSource()

```
public java.lang.Object getSource()
```

Return the stream whose NPT status changed.

#### Overrides:

[java.util.EventObject.getSource\(\)](#) in class [java.util.EventObject](#)

#### Returns:

the [DSMCCStream](#) whose status changed

# org.dvb.dsmcc ObjectChangeEvent

## Syntax

```
public class ObjectChangeEvent extends java.util.EventObject
```

```
java.lang.Object  
|  
+--java.util.EventObject  
|  
+--org.dvb.dsmcc.ObjectChangeEvent
```

## All Implemented Interfaces:

```
java.io.Serializable
```

## Description

This class describes an object change event that is used to monitor the arrival of a new version of a DSMCCObject. For files carried in a DSMCC object carousel, when a change in a module is detected, this event shall be sent to all registered listeners for all objects carried in that module.

## Constructors

---

### ObjectChangeEvent(DSMCCObject, int)

```
public ObjectChangeEvent(DSMCCObject source, int aVersionNumber)
```

Creates an ObjectChangeEvent indicating that a new version of the monitored DSMCC Object has been detected. It is up to the application to reload the new version of the object.

#### Parameters:

`source` - the DSMCCObject whose version has changed

`aVersionNumber` - the new version number.

## Methods

---

### getNewVersionNumber()

```
public int getNewVersionNumber()
```

This method is used to get the new version number of the monitored DSMCCObject. For files carried in a DSMCC object carousel, this method shall return the version number of the module carrying the file.

#### Returns:

the new version number.

---

### getSource()

```
public java.lang.Object getSource()
```

Returns the DSMCCObject that has changed

**Overrides:**

java.util.EventObject.getSource() in class java.util.EventObject

**Returns:**

the DSMCCObject that has changed

org.dvb.dsmcc

# ObjectChangeListener

## Syntax

```
public interface ObjectChangeListener extends java.util.EventListener
```

## All Superinterfaces:

java.util.EventListener

## Description

The objects that implements the ObjectChangeListener interface can receive ObjectChangeEvent event.

## Methods

---

### receiveObjectChangeEvent(ObjectChangeEvent)

```
public void receiveObjectChangeEvent (ObjectChangeEvent e)
```

Send a ObjectChangeEvent to the ObjectChangeListener.

#### Parameters:

e - the ObjectChangeEvent event.



# org.dvb.dsmcc ServerDeliveryErrorEvent

## Syntax

```
public class ServerDeliveryErrorEvent extends AsynchronousLoadingEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--AsynchronousLoadingEvent
|
+--org.dvb.dsmcc.ServerDeliveryErrorEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

The local machine can not communicate with the server. This event is only used with files implemented by delivery over bi-directional IP connections. For the object carousel the `MPEGDeliveryErrorEvent` is used instead.

## Constructors

---

### ServerDeliveryErrorEvent(DSMCCObject)

```
public ServerDeliveryErrorEvent(DSMCCObject o)
```

Creates a `ServerDeliveryErrorEvent` object.

#### Parameters:

o - the `DSMCCObject` that generated the event.

## Methods

---

### getSource()

```
public java.lang.Object getSource()
```

Returns the `DSMCCObject` that generated the event.

#### Overrides:

`getSource()` in class `AsynchronousLoadingEvent`

#### Returns:

the `DSMCCObject` that generated the event.

# org.dvb.dsmcc ServerDeliveryException

## Syntax

```
public class ServerDeliveryException extends DSMCCEException
```

```
java.lang.Object
|
+--java.lang.Throwable
|
+--java.lang.Exception
|
+--java.io.IOException
|
+--DSMCCEException
|
+--org.dvb.dsmcc.ServerDeliveryException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

A ServerDeliveryException is thrown when the local machine can not communicate with the server. This exception is only used with files implemented by delivery over a bi-directional IP connection. For the object carousel the MPEGDeliveryException is used instead.

## Constructors

---

### ServerDeliveryException()

```
public ServerDeliveryException()
```

Construct a ServerDeliveryException with no detail message

---

### ServerDeliveryException(String)

```
public ServerDeliveryException(java.lang.String s)
```

Construct a ServerDeliveryException with the specified detail message

#### Parameters:

s - the detail message

# org.dvb.dsmcc ServiceDomain

## Syntax

```
public class ServiceDomain
{
    java.lang.Object
    |
    +--org.dvb.dsmcc.ServiceDomain
}
```

## Description

A `ServiceDomain` represents a group of DSMCC objects. The objects are sent either using the object carousel for a broadcast network or with the DSM-CC User-to-User protocol for an interactive network.

To access the objects of a `ServiceDomain`, it has to be attached to the file system name space of the MHP terminal. To access the content of an object, the application has four ways:

- It can instantiate the class that is used to read the object (`java.io.FileInputStream` or `java.io.RandomAccessFile` for a File or `DSMCCStream` for a stream) from its pathname. The loading of the object is implicit but the application has no way to abort it. NB: Obviously, for the Object Carousel, the write mode of `java.io.RandomAccessFile` is not allowed.
- It can instantiate a `DSMCCObject` and carry out a Synchronous loading. The loading can be aborted by the `abort` method of the `DSMCCObject` class. When the object is loaded, the application will instantiate the class used to read the object.
- It can instantiate a `DSMCCObject` and carry out an Asynchronous loading. So several loading can be started in parallel from the same thread.
- It is also possible to create directly a `java.io.File` for a DSMCC object.

Instances of `ServiceDomain` exist in two states, attached and detached. Newly created instances are always in the detached state. They become attached when a call to the `attach` method succeeds. They become detached following a call to the `detach` method.

When service domains in the attached state temporarily loose their network connection, (e.g. if the MHP terminal tunes away from the transport stream where they are carried), the behaviour of DSMCC objects which are part of the service domain is specified in the main body of this specification. If such a network connection becomes available again then the service domain shall resume normal behaviour.

A service domain which is temporarily lost its network connection may be forced into the detached state by the implementation if the loss of the network connection becomes irrecoverable. The precise details of when this happens are implementation dependent. This is the only situation when shall be forced into the detached state. Once a `ServiceDomain` is detached, it will never be automatically attached.

## Constructors

---

### ServiceDomain()

```
public ServiceDomain()
{
    Creates a ServiceDomain object.
}
```

## Methods

---

**attach(byte[])**

```
public void attach(byte[] NSAPAddress)
```

This function is used to attach a `ServiceDomain` from either an object carousel or from an interactive network. This call will block until the attachment is done.

**Parameters:**

`NSAPAddress` - The NSAP Address of a `ServiceDomain` as defined in in ISO/IEC 13818-6

**Throws:**

`InterruptedException` - The attachment has been aborted.

`InvalidAddressException` - The NSAP Address is invalid.

`DSMCCEException` - An error has occurred during the attachment.

`MPEGDeliveryException` - attaching to this domain would require tuning.

---

**attach(Locator)**

```
public void attach(org.davic.net.Locator l)
```

This function is used to get a `ServiceDomain` from an object carousel. It loads the module which contains the service gateway object and mounts the `ServiceDomain` volume in the file system hierarchy. This call will block until the service gateway is loaded. It can be aborted by another thread with the method `detach`. In this case an `InterruptedException` is thrown.

**Parameters:**

`l` - The locator pointing to the elementary stream carrying the DSI of the object carousel, or to a DVB service that carries one and only one object carousel.

**Throws:**

`DSMCCEException` - An error has occurred during the attachment. For example, the locator does not point to a component carrying a DSI of an Object Carousel or to a service containing a single carousel)

`InterruptedException` - The attachment has been aborted.

`MPEGDeliveryException` - attaching to this domain would require tuning.

---

**attach(Locator, int)**

```
public void attach(org.davic.net.Locator aDVBSERVICE, int aCarouselId)
```

This function is used to get a `ServiceDomain` from an object carousel. It loads the module which contains the service gateway object and mounts the `ServiceDomain` volume in the file system hierarchy. This call will block until the service gateway is loaded. It can be aborted by another thread with the method `detach`. In this case an `InterruptedException` is thrown.

**Parameters:**

`aDVBSERVICE` - The coordinates of the DVB service which contains the object carousel. This locator has to point to a DVB service.

`aCarouselId` - The identifier of the carousel.

**Throws:**

`InterruptedException` - The attachment has been aborted.

`MPEGDeliveryException` - An MPEG error occurred (such as time-out).

`ServiceXFRException` - The service gateway cannot be loaded in the current service domain

---

**detach()**

```
public void detach()
```

A call to this method is a hint that the applications gives to the MHP to unmount the volume and delete the objects of the service domain. When another application is using objects of the same service domain the method has no effects. When there are no other application using objects of the service domain, a call to this method is a hint that the MHP can free all the resources allocated to this service domain.

After this, the `ServiceDomain` will be in a non-attached state and will behave as if it had just been constructed. Subsequent calls to `detach` shall throw `NotLoadedException`.

**Throws:**

`NotLoadedException` - is thrown if the `ServiceDomain` is not attached or if there is no call to `attach` in progress.

---

**getLocator()**

```
public org.davic.net.Locator getLocator()
```

Return the locator for this service domain. If this `ServiceDomain` instance was last attached by specifying a locator then that exact same locator shall be returned. If this `ServiceDomain` instance was last attached by specifying an NSAP address then the locator shall be generated from that address. If this `ServiceDomain` has never been attached then null shall be returned.

**Returns:**

a locator for this service domain

**Since:**

MHP 1.0.1

---

**getMountPoint()**

```
public DSMCCObject getMountPoint()
```

Returns a `DSMCCObject` object describing the top level directory of this `ServiceDomain`. If the `ServiceDomain` object is not attached then null is returned.

**Returns:**

an instance of `org.dvb.dsmcc.DSMCCObject` if mounted or null otherwise

**Since:**

MHP 1.0.1

---

**getNSAPAddress()**

```
public byte[] getNSAPAddress()
```

This method returns the NSAP address of the `ServiceDomain`.

**Returns:**

the NSAP address of the `ServiceDomain`.

**Throws:**

`NotLoadedException` - is thrown if the `ServiceDomain` is not attached.

---

**getURL(Locator)**

```
public static java.net.URL getURL(org.davic.net.Locator l)
```

Obtain a `java.net.URL` corresponding to a 'dvb:' locator. If the service domain corresponding to the locator is attached and the file referenced in the locator exists then an instance of `java.net.URL` is returned which can be used to reference this file.

**Parameters:**

1 - a locator object encapsulating a 'dvb:' locator which includes a 'dvb\_abs\_path' element.

**Returns:**

a `java.net.URL` which can be used to access the file referenced by the 'dvb:' locator

**Throws:**

`InvalidLocatorException` - if the locator is not a valid 'dvb:' locator or does not includes all elements including 'dvb\_abs\_path' element

`NotLoadedException` - is thrown if the locator is valid and includes enough information but it references a service domain which is not attached.

`FileNotFoundException` - if the service domain is attached but the file referenced by the locator does not exist

---

**isAttached()**

```
public boolean isAttached()
```

Return whether this service domain is in the attached or detached state.

**Returns:**

true if this service domain is in the attached state, otherwise false

**Since:**

MHP 1.0.1

---

**isNetworkConnectionAvailable()**

```
public boolean isNetworkConnectionAvailable()
```

Return whether the network connection for this service domain is available. This return value is independent of whether the service domain is attached or not. If a service domain is distributed across multiple network connections (e.g. using the optional support for DSMCC over IIOp) then this will reflect the availability of the network connection carrying the object mounted to the mount point.

**Returns:**

true if the network connection for this service domain is available otherwise false

**Since:**

MHP 1.0.1

# org.dvb.dsmcc ServiceXFRErrorEvent

## Syntax

```
public class ServiceXFRErrorEvent extends AsynchronousLoadingEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--AsynchronousLoadingEvent
|
+--org.dvb.dsmcc.ServiceXFRErrorEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

The object requested is available in an alternate ServiceDomain. When an application attempts to asynchronously load an object that has itself a LiteOptionProfileBody IOR or that has a parent directory that has a LiteOptionProfileBody IOR, this event shall be sent to the application. There is no implicit mounting by the implementation of the carousel that actually contains the object. This event is also sent even if the Service Domain that actually contains the DSMCCObject is already mounted.

## Constructors

---

### ServiceXFRErrorEvent(DSMCCObject, ServiceXFRReference)

```
public ServiceXFRErrorEvent(DSMCCObject o, ServiceXFRReference ref)
```

Creates a ServiceXFRErrorEvent object.

#### Parameters:

- o - the DSMCCObject that generated the event.
- ref - the address of an alternate ServiceDomain where the object can be found.

## Methods

---

### getServiceXFR()

```
public ServiceXFRReference getServiceXFR()
```

This method is used to get a reference to the service domain that contains the requested object.

#### Returns:

The address of an alternate ServiceDomain where the object can be found.

---

### getSource()

```
public java.lang.Object getSource()
```

Returns the DSMCCObject that generated the event.

#### Overrides:

`getSource()` in class `AsynchronousLoadingEvent`

#### Returns:

the DSMCCObject that generated the event.

# org.dvb.dsmcc ServiceXFRException

## Syntax

```
public class ServiceXFRException extends DSMCCException
```

```
java.lang.Object
|
+--java.lang.Throwable
|
+--java.lang.Exception
|
+--java.io.IOException
|
+--DSMCCException
|
+--org.dvb.dsmcc.ServiceXFRException
```

## All Implemented Interfaces:

```
java.io.Serializable
```

## Description

A ServiceXFRException is thrown when a DSMCC Object can not be loaded in the current ServiceDomain but is available in an alternate ServiceDomain (i.e. for an object Carousel, the IOR of the object or one of its parent directories contains a Lite Option Profile Body). There is no implicit mounting by the implementation of the carousel that actually contain the object. This exception is also thrown even if the Service Domain that actually contains the DSMCCObject is already mounted.

## Constructors

---

### ServiceXFRException(byte[], String)

```
public ServiceXFRException(byte[] NSAPAddress, java.lang.String pathName)
```

Creates a ServiceXFRException object.

#### Parameters:

NSAPAddress - The NSAP Address of a ServiceDomain as defined in ISO/IEC 13818-6

pathName - pathName of the object in the alternate ServiceDomain

---

### ServiceXFRException(Locator, int, String)

```
public ServiceXFRException(org.davic.net.Locator aService, int carouselId,
    java.lang.String pathName)
```

Creates a ServiceXFRException object.

#### Parameters:

aService - Locator of the Service

carouselId - Carousel Identifier

pathName - pathName of the object in the alternate ServiceDomain



## Methods

---

### **getServiceXFR()**

```
public ServiceXFRReference getServiceXFR()
```

This method is used to get the alternate ServiceDomain which contains the object requested.

**Returns:**

the address of an alternate ServiceDomain where the object can be found.

# org.dvb.dsmcc ServiceXFRReference

## Syntax

```
public class ServiceXFRReference  
  
java.lang.Object  
|  
+--org.dvb.dsmcc.ServiceXFRReference
```

## Description

A ServiceXFRReference object is used when a DSMCC Object can not be loaded in the current ServiceDomain but is available in an alternate ServiceDomain. Instances of this class are just containers. The parameters passed are merely stored and returned by the access methods. It is the responsibility of the platform when generating instances to use correct values.

## Constructors

---

### ServiceXFRReference(byte[], String)

```
public ServiceXFRReference(byte[] nsapAddress, java.lang.String pathName)
```

Creates a ServiceXFRReference object.

#### Parameters:

`nsapAddress` - The NSAP Address of a ServiceDomain as defined in ISO/IEC 13818-6  
`pathName` - pathName of the object in the alternate ServiceDomain

---

### ServiceXFRReference(Locator, int, String)

```
public ServiceXFRReference(org.davic.net.Locator serviceLocator, int carouselId,  
                           java.lang.String pathName)
```

Creates a ServiceXFRReference object.

#### Parameters:

`serviceLocator` - Locator of the Service  
`carouselId` - Carousel Identifier  
`pathName` - pathName of the object in the alternate ServiceDomain

## Methods

---

### getCarouselId()

```
public int getCarouselId()
```

This method returns the carousel identifier. If the object was constructed using the constructor which includes a carousel ID or if it was constructed using the constructor which includes an NSAP

address and that NSAP address contains a carouselID then this method shall return that carousel ID otherwise this method shall return -1.

**Returns:**

the carousel identifier or -1.

---

**getLocator()**

```
public org.davic.net.Locator getLocator()
```

This method returns the Locator of the Service for an Object Carousel.

**Returns:**

the Locator of the Service for an Object Carousel. This method returns null, if the ServiceDomain is not associated with an Object Carousel. In this case the NSAP address must be used instead.

---

**getNSAPAddress()**

```
public byte[] getNSAPAddress()
```

This method returns the NSAP Address of a ServiceDomain as defined in ISO/IEC 13818-6. If the object was constructed using an NSAP address then this method shall return the NSAP address passed into the constructor. If the object was constructed with a locator and a carouselID then this method shall return an NSAP address derived from this information when locator is an instance of org.davic.net.dvb.DVBLocator. Otherwise this method shall return null

**Returns:**

the NSAP Address of a ServiceDomain as defined in ISO/IEC 13818-6 or null

---

**getPathName()**

```
public java.lang.String getPathName()
```

This method returns the pathname of the object in the alternate ServiceDomain.

**Returns:**

the pathname of the object in the alternate ServiceDomain.

# org.dvb.dsmcc StreamEvent

## Syntax

```
public class StreamEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.dvb.dsmcc.StreamEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This class describes a Stream event which is used to synchronize an application with an MPEG Stream.

## Constructors

---

### StreamEvent(DSMCCStreamEvent, long, String, int, byte[])

```
public StreamEvent(DSMCCStreamEvent source, long npt, java.lang.String name, int eventId,
                  byte[] eventData)
```

Creates a StreamEvent object.

#### Parameters:

`source` - The DSMCCStreamEvent that has generated the event.

`npt` - The value of the NPT (Normal Play Time) when the event is triggered. This value is equal to the field eventNPT in the DSMCC StreamEventDescriptor except where the event is a "do it now" event in which case the value -1 is returned (as the value of NPT may not be meaningful).

`name` - The name of this event. The list of event names is located in the DSMCC StreamEvent object. This list is returned by the method DSMCCStreamEvent.getEventList.

`eventId` - The eventId of this event. The list of event IDs is located in the DSMCC StreamEvent object.

`eventData` - The application specific data found in the DSMCC StreamEventDescriptor.

## Methods

---

### getEventData()

```
public byte[] getEventData()
```

This method is used to retrieve the private data associated with the event.

#### Returns:

The private data associated with the event.

---

**getEventId()**

```
public int getEventId()
```

This method is used to get the identifier of the StreamEvent.

**Returns:**

The identifier of the StreamEvent.

---

**getEventName()**

```
public java.lang.String getEventName()
```

This method is used to get the name of the StreamEvent

**Returns:**

the name of the StreamEvent

---

**getEventNPT()**

```
public long getEventNPT()
```

This method is used to get the NPT of the Event in milliseconds.

**Returns:**

The NPT of the Event in milliseconds.

---

**getSource()**

```
public java.lang.Object getSource()
```

This method returns the DSMCCStreamEvent that generated the event.

**Overrides:**

java.util.EventObject.getSource() in class java.util.EventObject

**Returns:**

the DSMCCStreamEvent that generated the event.

org.dvb.dsmcc

# StreamEventListener

## Syntax

```
public interface StreamEventListener extends java.util.EventListener
```

## All Superinterfaces:

java.util.EventListener

## Description

Objects that implement the StreamEventListener interface can receive StreamEvent event.

## Methods

---

### receiveStreamEvent(StreamEvent)

```
public void receiveStreamEvent(StreamEvent e)
```

Send a StreamEvent to the StreamEventListener.

#### Parameters:

e - the StreamEvent event.

# org.dvb.dsmcc SuccessEvent

## Syntax

```
public class SuccessEvent extends AsynchronousLoadingEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--AsynchronousLoadingEvent
|
+--org.dvb.dsmcc.SuccessEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This event indicates that the asynchronous loading was successful.

## Constructors

---

### SuccessEvent(DSMCCObject)

```
public SuccessEvent(DSMCCObject o)
```

Creates a SuccessEvent object.

**Parameters:**

o - the DSMCCObject which was successfully loaded.

## Methods

---

### getSource()

```
public java.lang.Object getSource()
```

Returns the DSMCCObject which was successfully loaded.

**Overrides:**

getSource() in class AsynchronousLoadingEvent

**Returns:**

the loaded DSMCCObject

# org.dvb.dsmcc UnknownEventException

## Syntax

```
public class UnknownEventException extends DSMCCException
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--java.io.IOException
            |
            +--DSMCCException
                |
                +--org.dvb.dsmcc.UnknownEventException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

The `UnknownEventException` is thrown when a method tries to access to an unknown event. This exception may get thrown because the event in question is not being signalled yet. It does not indicate that the event is permanently unavailable. Applications may choose to attempt to subscribe to the event again at a later point in time in the expectation that the event has become available since the previous attempt.

## Constructors

---

### UnknownEventException()

```
public UnknownEventException()
```

Construct an `UnknownEventException` with no detail message

---

### UnknownEventException(String)

```
public UnknownEventException(java.lang.String s)
```

Construct an `UnknownEventException` with the specified detail message

#### Parameters:

`s` - the detail message



---

## **Annex Q (normative): Datagram Socket Buffer Control**

# Package org.dvb.net

## Description

Provides general networking features not found elsewhere.

### Class Summary

#### Classes

<code>DatagramSocketBuffer-</code> <code>Control</code>	This class provides additional control over buffering for <code>DatagramSockets</code> .
--	--

org.dvb.net

# DatagramSocketBufferControl

## Syntax

```
public class DatagramSocketBufferControl
    java.lang.Object
    |
    +--org.dvb.net.DatagramSocketBufferControl
```

## Description

This class provides additional control over buffering for `DatagramSockets`.

## Methods

---

### **getReceiveBufferSize(DatagramSocket)**

```
public static int getReceiveBufferSize(java.net.DatagramSocket d)
```

Get value of the `SO_RCVBUF` option for this socket, that is the buffer size used by the platform for input on the this Socket.

#### **Parameters:**

`d` - The `DatagramSocket` for which to query the receive buffer size.

#### **Returns:**

The size of the receive buffer, in bytes.

#### **Throws:**

`SocketException` - - If there is an error when querying the `SO_RCVBUF` option.

---

### **setReceiveBufferSize(DatagramSocket, int)**

```
public static void setReceiveBufferSize(java.net.DatagramSocket d, int size)
```

Sets the `SO_RCVBUF` option to the specified value for this `DatagramSocket`. The `SO_RCVBUF` option is used by the platform's networking code as a hint for the size to use when allocating the underlying network I/O buffers.

Increasing buffer size can increase the performance of network I/O for high-volume connection, while decreasing it can help reduce the backlog of incoming data. For UDP, this sets the buffer size for received packets.

Because `SO_RCVBUF` is a hint, applications that want to verify what size the buffers were set to should call `getReceiveBufferSize`. This method shall throw `IllegalArgumentException` - if size is 0 or is negative.

#### **Parameters:**

`d` - The `DatagramSocket` for which to change the receive buffer size.

`size` - The requested size of the receive buffer, in bytes.

#### **Throws:**

`SocketException` - - If there is an error when setting the `SO_RCVBUF` option.

---

## **Annex R (normative): DVB-J Return Channel Connection Management API**

# Package org.dvb.net.rc

## Description

Provides session management for bi-directional IP connections which are session based from the point of view of an application. The best example of this is a conventional modem.

Class Summary	
<b>Interfaces</b>	
<code>ConnectionListener</code>	This interface should be implemented by objects wishing to be notified about the connection status of a <code>ConnectionRCInterface</code> .
<b>Classes</b>	
<code>ConnectionEstablishedEvent</code>	<code>ConnectionEstablishedEvent</code> - An event generated after a connection is established for a <code>ConnectionRCInterface</code> .
<code>ConnectionFailedEvent</code>	<code>ConnectionFailedEvent</code> - An event generated after an attempt to setup a connection for a <code>ConnectionRCInterface</code> fails.
<code>ConnectionParameters</code>	This class encapsulates the parameters needed to specify the target of a connection.
<code>ConnectionRCEvent</code>	<code>ConnectionRCEvent</code> - the base class for events related to connection oriented return channels.
<code>ConnectionRCInterface</code>	This class models a connection based return channel network interface for use in receiving and transmitting IP packets over a return channel.
<code>ConnectionTerminatedEvent</code>	<code>ConnectionTerminatedEvent</code> - An event generated after a connected <code>ConnectionRCInterface</code> is disconnected.
<code>RCInterface</code>	This class models a return channel network interface for use in receiving and transmitting IP packets over a logical return channel.
<code>RCInterfaceManager</code>	This class is the factory and manager for all return channel interfaces in the system.
<code>RCInterfaceReleasedEvent</code>	This event informs an application that a <code>RCInterface</code> has been released by an application or other entity in the system.
<code>RCInterfaceReservedEvent</code>	This event informs an application that a <code>RCInterface</code> has been reserved by an application or other entity in the system.
<code>RCPermission</code>	This class is for return channel set-up permissions.
<b>Exceptions</b>	
<code>IncompleteTargetException</code>	Thrown when the target for a connection is incompletely specified.
<code>PermissionDeniedException</code>	Thrown when an application calls a method which it does not have permission to call at that time.

org.dvb.net.rc

# ConnectionEstablishedEvent

## Syntax

```
public class ConnectionEstablishedEvent extends ConnectionRCEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|   |
|   +--ConnectionRCEvent
|       |
|       +--org.dvb.net.rc.ConnectionEstablishedEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

ConnectionEstablishedEvent - An event generated after a connection is established for a ConnectionRCInterface.

## Constructors

---

### ConnectionEstablishedEvent(Object)

```
public ConnectionEstablishedEvent(java.lang.Object source)
```

Construct an event.

#### Parameters:

source - the ConnectionRCInterface whose connection was established

# org.dvb.net.rc ConnectionFailedEvent

## Syntax

```
public class ConnectionFailedEvent extends ConnectionRCEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--ConnectionRCEvent
|
+--org.dvb.net.rc.ConnectionFailedEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

ConnectionFailedEvent - An event generated after an attempt to setup a connection for a ConnectionRCInterface fails.

## Constructors

---

### ConnectionFailedEvent(Object)

```
public ConnectionFailedEvent(java.lang.Object source)
```

Construct an event.

#### Parameters:

source - the ConnectionRCInterface whose connection attempt failed

# org.dvb.net.rc ConnectionListener

## Syntax

```
public interface ConnectionListener
```

## Description

This interface should be implemented by objects wishing to be notified about the connection status of a `ConnectionRCInterface`.

## Methods

---

### **connectionChanged(ConnectionRCEvent)**

```
public void connectionChanged(ConnectionRCEvent e)
```

This method is called to report events related to the setup and termination of return channel interface connections.

#### **Parameters:**

e - the event which happened



org.dvb.net.rc

# ConnectionParameters

## Syntax

```
public class ConnectionParameters
    java.lang.Object
    |
    +--org.dvb.net.rc.ConnectionParameters
```

## Description

This class encapsulates the parameters needed to specify the target of a connection.

## Constructors

---

### ConnectionParameters(String, String, String)

```
public ConnectionParameters(java.lang.String number, java.lang.String username,
                             java.lang.String password)
```

Construct a set of connection parameters. Details of the DNS server to use are supplied by the server.

#### Parameters:

- `number` - the target of the connection, e.g. a phone number
- `username` - the username to use in connection setup
- `password` - the password to use in connection setup

---

### ConnectionParameters(String, String, String, InetAddress[])

```
public ConnectionParameters(java.lang.String number, java.lang.String username,
                             java.lang.String password, java.net.InetAddress[] dns)
```

Construct a set of connection parameters.

#### Parameters:

- `number` - the target of the connection, e.g. a phone number
- `username` - the username to use in connection setup
- `password` - the password to use in connection setup
- `dns` - the list of DNS servers to try before reporting failure. The order in which they are interrogated is not specified. Once one result has been obtained, there is no requirement to try others.

## Methods

---

### getDNSServer()

```
public java.net.InetAddress[] getDNSServer()
```

Return the addresses of the DNS servers to use for the connection

**Returns:**

return the addresses of the DNS servers to use for the connection or null where the DNS address is server supplied.

---

**getPassword()**

```
public java.lang.String getPassword()
```

Return the password used in establishing this connection

**Returns:**

the password used in establishing this connection

---

**getTarget()**

```
public java.lang.String getTarget()
```

Return the target of this connection for example a phone number

**Returns:**

the target of the connection

---

**getUsername()**

```
public java.lang.String getUsername()
```

Return the username used in establishing this connection

**Returns:**

the username used in establishing the connection

# org.dvb.net.rc ConnectionRCEvent

## Syntax

```
public class ConnectionRCEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.dvb.net.rc.ConnectionRCEvent
```

## Direct Known Subclasses:

```
ConnectionEstablishedEvent, ConnectionFailedEvent,
ConnectionTerminatedEvent
```

## All Implemented Interfaces:

```
java.io.Serializable
```

## Description

ConnectionRCEvent - the base class for events related to connection oriented return channels.

## Constructors

---

### ConnectionRCEvent(Object)

```
public ConnectionRCEvent(java.lang.Object source)
```

Construct an event

#### Parameters:

source - the ConnectionRCInterface for which the event was generated.

org.dvb.net.rc

# ConnectionRCInterface

## Syntax

```
public class ConnectionRCInterface extends RCInterface implements
    org.davic.resources.ResourceProxy
```

```
java.lang.Object
|
+--RCInterface
    |
    +--org.dvb.net.rc.ConnectionRCInterface
```

## All Implemented Interfaces:

org.davic.resources.ResourceProxy

## Description

This class models a connection based return channel network interface for use in receiving and transmitting IP packets over a return channel. Targets for connections are specified as strings including the number to dial. These strings can only include numbers and are passed direct to the underlying device.

## Methods

---

### addConnectionListener(ConnectionListener)

```
public void addConnectionListener(ConnectionListener l)
```

Add a listener for events related to connections of this interface.

#### Parameters:

`l` - the listener for the connection related events

---

### connect()

```
public void connect()
```

Connect this return channel to the current target. If this ResourceProxy does not have the underlying resource reserved then a `PermissionDeniedException` will be thrown. Where the underlying resource is reserved but at the time the method is called, it is known that connection is impossible then an `IOException` will be thrown. Apart from this, this method is asynchronous and completion or failure is reported through the event listener on this class. If a connection is already established when this method is called then the method call shall have no effect.

#### Throws:

`PermissionDeniedException` - if this application does not own the resource

`IOException` - if connection is known to be impossible at the time when the method is called

---

### disconnect()

```
public void disconnect()
```

Disconnect this return channel from the current target. This method is asynchronous and completion is reported through the event listener on this class. This method does not release the underlying resource from the ResourceProxy. If no connection is established then this method shall have no effect.

**Throws:**

`PermissionDeniedException` - if this application does not own the resource

---

**getClient()**

```
public org.davic.resources.ResourceClient getClient()
```

Return the object which asked to be notified about withdrawal of the underlying resource. This is the object provided as the first parameter to the last call to the reserve method on this object. If this object does not have the underlying resource reserved then null is returned.

**Specified By:**

`org.davic.resources.ResourceProxy.getClient()` in interface `org.davic.resources.ResourceProxy`

**Returns:**

the object which asked to be notified about withdrawal of the underlying physical resource from this resource proxy or null

---

**getConnectedTime()**

```
public int getConnectedTime()
```

Return the time an interface has been connected

**Returns:**

the time in seconds for which this interface has been connected or -1 if the device is not connected

---

**getCurrentTarget()**

```
public ConnectionParameters getCurrentTarget()
```

Get the current target for connections.

**Returns:**

the current set of connection target parameters

**Throws:**

`IncompleteTargetException` - if the current target is not completely configured

`SecurityException` - if the application is not allowed to read the current target as defined by the security policy of the platform

---

**getSetupTimeEstimate()**

```
public float getSetupTimeEstimate()
```

Obtain an estimate of the setup time for a successful connection for this interface in seconds.

**Returns:**

an estimate of the setup time for a successful connection for this interface in seconds.

---

**isConnected()**

```
public boolean isConnected()
```

Check if this interface is connected. Connected means able to receive and transmit packets.

**Returns:**

true if the interface is connected, otherwise false

---

**release()**

```
public void release()
```

Release the right to control this return channel interface. If this object does not have the right to control this return channel interface then this method shall have no effect.

---

**removeConnectionListener(ConnectionListener)**

```
public void removeConnectionListener(ConnectionListener l)
```

Remove a listener for events related to connections of this interface. If the listener specified is not currently receiving these events then this method has no effect.

**Parameters:**

l - the listener for the connection related events

---

**reserve(ResourceClient, Object)**

```
public void reserve(org.davic.resources.ResourceClient c, java.lang.Object requestData)
```

Request the right to control this return channel interface. If the right to control the return channel interface has already been reserved then this method shall have no effect.

**Parameters:**

c - the object to be notified when resources are removed

requestData - Used by the Resource Notification API in the requestRelease method of the ResourceClient interface. The usage of this parameter is optional and a null reference may be supplied.

**Throws:**

PermissionDeniedException - if this interface cannot be reserved

SecurityException - if the application is denied access to the resource by security policy.

---

**setTarget(ConnectionParameters)**

```
public void setTarget(ConnectionParameters target)
```

Set a non-default target for connections.

**Parameters:**

target - the new set of connection target parameters

**Throws:**

IncompleteTargetException - if the application owns the resource but the target is not completely specified

PermissionDeniedException - if this application does not own the resource

SecurityException - if the application is not allowed to modify the target as defined by the security policy of the platform

---

**setTargetToDefault()**

```
public void setTargetToDefault()
```

Set the target for connections to the default.

**Throws:**

`IncompleteTargetException` - if the application owns the resource but the target is not completely specified

`PermissionDeniedException` - if this application does not own the resource

`SecurityException` - if the application is not allowed to connect to the default target

org.dvb.net.rc

# ConnectionTerminatedEvent

## Syntax

```
public class ConnectionTerminatedEvent extends ConnectionRCEvent
```

```
java.lang.Object
|
+--java.util.EventObject
    |
    +--ConnectionRCEvent
        |
        +--org.dvb.net.rc.ConnectionTerminatedEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

ConnectionTerminatedEvent - An event generated after a connected ConnectionRCInterface is disconnected.

## Constructors

---

### ConnectionTerminatedEvent(Object)

```
public ConnectionTerminatedEvent(java.lang.Object source)
```

Construct an event.

#### Parameters:

source - the ConnectionRCInterface whose status changed



org.dvb.net.rc

# IncompleteTargetException

## Syntax

```
public class IncompleteTargetException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
   |
   +--java.lang.Exception
      |
      +--org.dvb.net.rc.IncompleteTargetException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Thrown when the target for a connection is incompletely specified. This is thrown either when the default connection target is incompletely defined in the device or when an application provides an incompletely defined connection target to the device or when the connection target is badly formed, e.g. includes illegal characters in a number parameter.

## Constructors

---

### IncompleteTargetException()

```
public IncompleteTargetException()
```

Default constructor for the exception

---

### IncompleteTargetException(String)

```
public IncompleteTargetException(java.lang.String reason)
```

Constructor for the exception with a specified reason

#### Parameters:

`reason` - the reason why the exception was raised

# org.dvb.net.rc PermissionDeniedException

## Syntax

```
public class PermissionDeniedException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
   |
   +--java.lang.Exception
      |
      +--org.dvb.net.rc.PermissionDeniedException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Thrown when an application calls a method which it does not have permission to call at that time.

## Constructors

---

### PermissionDeniedException()

```
public PermissionDeniedException()
```

Default constructor for the exception

---

### PermissionDeniedException(String)

```
public PermissionDeniedException(java.lang.String reason)
```

Constructor for the exception with a specified reason

#### Parameters:

`reason` - the reason why the exception was raised

# org.dvb.net.rc RCInterface

## Syntax

```
public class RCInterface
{
    java.lang.Object
    |
    +--org.dvb.net.rc.RCInterface
}
```

## Direct Known Subclasses:

[ConnectionRCInterface](#)

## Description

This class models a return channel network interface for use in receiving and transmitting IP packets over a logical return channel. This can include real analog modems, cable return channel and all the other options allowed by the relevant DVB specification. This class does not model any concept of connection. Hence interfaces represented by this class and not by a sub-class of it are permanently connected.

## Fields

---

### TYPE\_CATV

```
public static final int TYPE_CATV
Constant to indicate a CATV return channel.
```

---

### TYPE\_DECT

```
public static final int TYPE_DECT
Constant to indicate a DECT return channel.
```

---

### TYPE\_ISDN

```
public static final int TYPE_ISDN
Constant to indicate an ISDN return channel.
```

---

### TYPE\_LMDS

```
public static final int TYPE_LMDS
Constant to indicate a LMDS return channel.
```

---

### TYPE\_MATV

```
public static final int TYPE_MATV
Constant to indicate a MATV return channel.
```

---

**TYPE\_PSTN**

```
public static final int TYPE_PSTN
```

Constant to indicate a PSTN return channel.

## Methods

---

**getDataRate()**

```
public int getDataRate()
```

Return the maximum data rate of the connection over the immediate access network to which this network interface is connected. For asymmetric connections, the data rate coming into the MHP terminal shall be returned. For connection oriented interfaces which are not currently connected, the value returned shall be that of the last connection established where that information is available. Where that information is not available, (e.g. where no connection has been established since an MHP terminal was power cycled), -1 shall be returned.

**Returns:**

a data rate in KBaud or -1 where this is not available

**Since:**

MHP 1.0.1

---

**getType()**

```
public int getType()
```

Return the type of return channel represented by this object.

**Returns:**

the type of return channel represented by this object encoded as one of the constants defined in this class

# org.dvb.net.rc RCInterfaceManager

## Syntax

```
public class RCInterfaceManager implements org.davic.resources.ResourceServer  
  
java.lang.Object  
|  
+--org.dvb.net.rc.RCInterfaceManager
```

## All Implemented Interfaces:

org.davic.resources.ResourceServer

## Description

This class is the factory and manager for all return channel interfaces in the system. The methods on this class which return instances of the `RCInterface` will only return new instances of that class under the following conditions:-

- on the first occasion an instance needs to be returned to a particular application for a particular interface.
- when new return channel interfaces are added to the system

## Methods

---

### addResourceStatusEventListener(ResourceStatusListener)

```
public void addResourceStatusEventListener(org.davic.resources.ResourceStatusListener  
listener)
```

This method informs a resource server that a particular object should be informed of changes in the state of the resources managed by that server.

#### Specified By:

org.davic.resources.ResourceServer.addResourceStatusEventListener(org.davic.resources.ResourceStatusListener) in interface org.davic.resources.ResourceServer

#### Parameters:

listener - the object to be informed of state changes

---

### getInstance()

```
public static RCInterfaceManager getInstance()
```

Factory method to obtain a manager. From the point of view of any individual application, this is a singleton.

#### Returns:

an instance of an `RCInterfaceManager`

---

### getInterface(InetAddress)

```
public RCInterface getInterface(java.net.InetAddress addr)
```

Return the interface which will be used when connecting to a particular host. Null is returned if this is not known when the method is called.

**Parameters:**

`addr` - the IP address of the host to connect to

**Returns:**

the interface which will be used or null if this is not known

---

**getInterface(Socket)**

```
public RCInterface getInterface(java.net.Socket s)
```

Return the interface which is used for a particular socket.

**Parameters:**

`s` - the socket to use

**Returns:**

the interface which is used or null if the socket isn't connected

---

**getInterface(URLConnection)**

```
public RCInterface getInterface(java.net.URLConnection u)
```

Return the interface which is used for a particular URLConnection

**Parameters:**

`u` - the URLConnection to use

**Returns:**

the interface which is used or null if the URLConnection isn't connected

---

**getInterfaces()**

```
public RCInterface[] getInterfaces()
```

Factory method to return a list of all return channel interfaces visible to this application. The number of entries in the array will exactly match the number of return channel interfaces visible to the application. Null is returned if no interfaces are visible to this application.

**Returns:**

an array of available return channel interfaces

---

**removeResourceStatusEventListener(ResourceStatusListener)**

```
public void removeResourceStatusEventListener(org.davic.resources.ResourceStatusListener listener)
```

This method informs a resource server that a particular object is no longer interested in being informed about changes in state of resources managed by that server. If the object had not registered its interest initially then this method has no effect.

**Specified By:**

`org.davic.resources.ResourceServer.removeResourceStatusEventListener(org.davic.resources.ResourceStatusListener)` in interface `org.davic.resources.ResourceServer`

**Parameters:**

`listener` - the object which is no longer interested

# org.dvb.net.rc RCInterfaceReleasedEvent

## Syntax

```
public class RCInterfaceReleasedEvent extends org.davic.resources.ResourceStatusEvent
```

```
java.lang.Object
|
|--org.davic.resources.ResourceStatusEvent
|
|--org.dvb.net.rc.RCInterfaceReleasedEvent
```

## Description

This event informs an application that a `RCInterface` has been released by an application or other entity in the system. It is generated when an application which had successfully reserved a `RCInterface` calls the `ConnectionRCInterface.release` method. It will also be generated if any other entities in the system own such an interface and then release that interface in such a way that it could then become available to applications using this API.

## Constructors

---

### RCInterfaceReleasedEvent(Object)

```
public RCInterfaceReleasedEvent(java.lang.Object bg)
```

Constructor for the event

#### Parameters:

`bg` - the `RCInterface` which has been released

## Methods

---

### getSource()

```
public java.lang.Object getSource()
```

Returns the device that has been released

#### Overrides:

`org.davic.resources.ResourceStatusEvent.getSource()` in class `org.davic.resources.ResourceStatusEvent`

#### Returns:

the `RCInterface` object representing the interface that has been released

# org.dvb.net.rc RCInterfaceReservedEvent

## Syntax

```
public class RCInterfaceReservedEvent extends org.davic.resources.ResourceStatusEvent
```

```
java.lang.Object
|
+--org.davic.resources.ResourceStatusEvent
|
+--org.dvb.net.rc.RCInterfaceReservedEvent
```

## Description

This event informs an application that a `RCInterface` has been reserved by an application or other entity in the system. It is generated when an application successfully reserves a `RCInterface`. It will also be generated if any other entities in the system reserve such an interface with the effect of something which was visible to applications using this API becoming unavailable.

## Constructors

---

### RCInterfaceReservedEvent(Object)

```
public RCInterfaceReservedEvent(java.lang.Object bg)
```

Constructor for the event

**Parameters:**

bg - the `RCInterface` representing the device which has been reserved

## Methods

---

### getSource()

```
public java.lang.Object getSource()
```

Returns the device that has been reserved

**Overrides:**

`org.davic.resources.ResourceStatusEvent.getSource()` in class  
`org.davic.resources.ResourceStatusEvent`

**Returns:**

an `RCInterface` representing the device that has been reserved



# org.dvb.net.rc RCPermission

## Syntax

```
public class RCPermission extends java.security.BasicPermission
```

```
java.lang.Object
|
+--java.security.Permission
|
+--java.security.BasicPermission
|
+--org.dvb.net.rc.RCPermission
```

## All Implemented Interfaces:

java.security.Guard, java.io.Serializable

## Description

This class is for return channel set-up permissions. An RCPermission contains a name, but no actions list.

The permission name can be "target:default", which indicates the permission to use the default connection parameters.

The permission name can also be "target:<phone number>", which indicates the permission to use the specified phone number in the connection set-up (ConnectionRCInterface.setTarget(ConnectionParameters) method).

A wildcard may be used at the end of the permission name. In that case, all phone numbers starting with the number before the wildcard are included in the permission.

Examples:

- target:0206234342 (Permission to dial the specified phone number)
- target:020\* (Permission to dial phone numbers starting with 020)
- target:\* (Permission to dial all phone numbers, including the default)

Note: ConnectionRCInterface.reserve(ResourceClient, Object) will throw a SecurityException if the application is not allowed to set-up a connection over the return channel at all (i.e., there is no valid target allowed).

## Constructors

---

### RCPermission(String)

```
public RCPermission(java.lang.String name)
```

Creates a new RCPermission with the specified name. The name is the symbolic name of the RCPermission.

#### Parameters:

name - the name of the RCPermission

---

**RCPPermission(String, String)**

```
public RCPPermission(java.lang.String name, java.lang.String actions)
```

Creates a new RCPPermission object with the specified name. The name is the symbolic name of the RCPPermission, and the actions String is unused and should be null. This constructor exists for use by the Policy object to instantiate new Permission objects.

**Parameters:**

`name` - the name of the RCPPermission

`actions` - should be null.

---

## **Annex S (normative): Application Listing and Launching**

# Package org.dvb.application

## Description

Provides access to lists of applications which are available in this context and the ability to launch those applications.

Class Summary	
<b>Interfaces</b>	
<code>AppAttributes</code>	The <code>AppAttributes</code> class is a mapping of various information about a registered application.
<code>AppProxy</code>	An <code>AppProxy</code> Object is a proxy to an application.
<code>AppsDatabaseEventListener</code>	The <code>AppsDatabaseListener</code> class allows an application to monitor the application database so that it can keep an up to date interface without polling the state.
<code>AppStateChangeListener</code>	The <code>AppStateChangeListener</code> class allows a launcher application to keep track of applications it launches or other applications running as part of the same service.
<code>DVBHTMLProxy</code>	A <code>DVBHTMLProxy</code> Object is a proxy to a DVBHTML application.
<code>DVBJSProxy</code>	A <code>DVBJSProxy</code> Object is a proxy to a DVBJS application.
<b>Classes</b>	
<code>AppIcon</code>	The <code>AppIcon</code> encapsulates the information concerning the icon attached to the application
<code>AppID</code>	The <code>AppID</code> is a representation of the unique identifier for applications.
<code>AppsControlPermission</code>	This class represents a Permission to control the lifecycle of another application.
<code>AppsDatabase</code>	The <code>AppsDatabase</code> is an abstract view of the currently available applications.
<code>AppsDatabaseEvent</code>	The <code>AppsDatabaseEvent</code> class indicates either the an entry in the application database has changed, or so many changes have occurred.
<code>AppsDatabaseFilter</code>	Abstract class for the filters.
<code>AppStateChangeEvent</code>	The <code>AppStateChangeEvent</code> class indicates a state transition of the application.
<code>CurrentServiceFilter</code>	Instances of <code>CurrentServiceFilter</code> are used to set a filter on the list of applications that are retrieved from the <code>AppsDatabase</code> (See methods <code>getAppAttributes</code> and <code>getAppIDs</code> )
<b>Exceptions</b>	
<code>IllegalProfileParameterException</code>	The <code>IllegalProfileParameter</code> exception is thrown if the application attempts to ask for a version number for a profile not specified for the application.
<code>LanguageNotAvailableException</code>	The <code>LanguageNotAvailableException</code> exception is thrown if the application asks for the name of an application in a language not signalled in the AIT.

# org.dvb.application

# AppAttributes

## Syntax

```
public interface AppAttributes
```

## Description

The `AppAttributes` class is a mapping of various information about a registered application. For applications which are signalled in an AIT, the mapping between the values returned by methods in this class and the fields and descriptors of the AIT shall be as specified in the main body of this specification.

Instances of objects implementing this interface are immutable and populated before the instance is first returned to an application.

## Since:

MHP1.0

## Fields

---

### DVB\_HTML\_application

```
public static final int DVB_HTML_application
```

The DVB registered value for all DVB-HTML applications.

---

### DVB\_J\_application

```
public static final int DVB_J_application
```

The DVB registered value for all DVB-J applications.

## Methods

---

### getAppIcon()

```
public AppIcon getAppIcon()
```

This method returns an object encapsulating the information about the icon(s) for the application.

#### Returns:

the information related to the icons that are attached to the application or null if no icon information is available

#### Since:

MHP1.0

---

### getIdentifier()

```
public AppID getIdentifier()
```

This method returns the application identifier. depending on the

**Returns:**

the application identifier

**Since:**

MHP1.0

---

**getIsServiceBound()**

```
public boolean getIsServiceBound()
```

This method determines whether the application is bound to a single service.

**Returns:**

true if the application is bound to a single service, false otherwise.

**Since:**

MHP1.0

---

**getName()**

```
public java.lang.String getName()
```

This method returns the name of the application. If the default language (as specified in user preferences) is in the set of available language / name pairs then the name in that language shall be returned. Otherwise this method will return a name which appears in that set on a "best-effort basis".

**Returns:**

the name of the application

**Since:**

MHP1.0

---

**getName(String)**

```
public java.lang.String getName(java.lang.String iso639code)
```

This method returns the name of the application in the language which is specified by the parameter passed as an argument. If the language specified is not in the set of available language /name pairs then an exception shall be thrown.

**Parameters:**

`iso639code` - the specified language, encoded as per ISO 639.

**Returns:**

returns the name of the application in the specified language

**Throws:**

[LanguageNotAvailableException](#) - if the name is not available in the language specified

**Since:**

MHP1.0

---

**getNames()**

```
public java.lang.String[][] getNames()
```

This method returns all the available names for the application together with their ISO 639 language code.

**Returns:**

the possible names of the application, along with their ISO 639 language code. The first string in each sub-array is the ISO 639 language code. The second string in each sub-array is the corresponding application name.

**Since:**

MHP1.0

---

**getPriority()**

```
public int getPriority()
```

This method returns the priority of the application.

**Returns:**

the priority of the application.

**Since:**

MHP1.0

---

**getProfiles()**

```
public java.lang.String[] getProfiles()
```

This method returns those minimum profiles required for the application may execute. Profile names shall be encoded using the same encoding specified elsewhere in this specification as input for use with the `java.lang.System.getProperty` method to query if a profile is supported by this platform.

For example, for implementations conforming to the first version of the specification, the translation from AIT signaling values to strings shall be as follows:

- '1' in the signaling will be translated into 'mhp.profile.enhanced\_broadcast'
- '2' in the signaling will be translated into 'mhp.profile.interactive\_broadcast'

Only profiles known to this particular MHP terminal shall be returned. Hence the method can return an array of size zero where all the profiles on which an application can execute an application are unknown.

**Returns:**

an array of Strings, each String describing a profile.

**Since:**

MHP1.0

---

**getProperty(String)**

```
public java.lang.Object getProperty(java.lang.String index)
```

The following method is included for properties that do not have explicit property accessors. The naming of properties and their return values are described in the main body of this specification.

**Parameters:**

`index` - a property name

**Returns:**

either the return value corresponding to the property name or null if the property name is unknown

**Since:**

MHP1.0

---

**getServiceLocator()**

```
public org.davic.net.Locator getServiceLocator()
```

This method returns the locator of the Service describing the application. For an application transmitted on a remote connection, the returned locator shall be the service for that remote connection. For applications not transmitted on a remote connection, the service returned shall be the currently selected service of the service context within which the application calling the method is running.

**Returns:**

the locator of the Service describing the application.

**Since:**

MHP1.0

---

**getType()**

```
public int getType()
```

This method returns the type of the application ( as registered by DVB).

**Returns:**

the type of the application ( as registered by DVB).

**Since:**

MHP1.0

---

**getVersions(String)**

```
public int[] getVersions(java.lang.String profile)
```

This method returns an array of integers containing the version number of the specification required to run this application at the specified profile.

**Parameters:**

`profile` - a profile encoded as described in the main body of this specification for use with `java.lang.System.getProperty`.

**Returns:**

an array of integers, containing the major, minor and micro values (in that order) required for the specified profile.

**Throws:**

`IllegalProfileParameterException` - thrown if the profile specified is not one of the minimum profiles required for the application to execute.

**Since:**

MHP1.0

---

**isStartable()**

```
public boolean isStartable()
```

This method determines whether the application is startable or not. An Application is not startable if any of the following apply.

- The application is transmitted on a remote connection.
- The caller of the method does not have the Permissions to start it.
- At the moment when the method is called, the implementation has detected that this application is not available any more.



The value returned by this method does not depend on whether the application is actually running or not.

**Returns:**

true if an application is startable, false otherwise.

**Since:**

MHP1.0

# org.dvb.application AppIcon

## Syntax

```
public class AppIcon
```

```
java.lang.Object
```

```
|
```

```
+--org.dvb.application.AppIcon
```

## Description

The `AppIcon` encapsulates the information concerning the icon attached to the application

## Constructors

---

### AppIcon()

```
protected AppIcon()
```

The constructor for the class. This constructor is intended for implementation convenience and evolution of the specification and not for use by MHP applications. Applications should obtain instances of this class from `AppAttributes.getAppIcon`.

**See Also:**

[getAppIcon\(\)](#)

## Methods

---

### getIconFlags()

```
public java.util.BitSet getIconFlags()
```

This method returns the flags identifying which icons are provided for the application.

**Returns:**

the icon flags encoded as a `BitSet`

**Since:**

MHP1.0

---

### getLocator()

```
public org.davic.net.Locator getLocator()
```

This method returns the location of the directory containing the application icons.

**Returns:**

the location of the directory containing the application icons.

**Since:**

MHP1.0

# org.dvb.application AppID

## Syntax

```
public class AppID
    java.lang.Object
    |
    +--org.dvb.application.AppID
```

## Description

The `AppID` is a representation of the unique identifier for applications.

Its string form is the Hex representation of the 48 bit number.

## Constructors

---

### AppID(int, int)

```
public AppID(int oid, int aid)
```

Create a new `AppID` based on the given integers. There is no range checking on these numbers.

**Parameters:**

`oid` - the globally unique organization number.

`aid` - the unique count within the organization.

**Since:**

MHP1.0

## Methods

---

### getAID()

```
public int getAID()
```

This method returns the integer value of the application count supplied in the constructor

**Returns:**

the integer value of the application count supplied in the constructor

**Since:**

MHP1.0

---

### getOID()

```
public int getOID()
```

This method returns the integer value of the organization number supplied in the constructor.

**Returns:**

the integer value of the organization number supplied in the constructor.

**Since:**  
MHP1.0

---

**toString()**

```
public java.lang.String toString()
```

This method returns a string containing the Hex representation of the 48 bit number.

**Overrides:**  
java.lang.Object.toString() in class java.lang.Object

**Returns:**  
a string containing the Hex representation of the 48 bit number.

**Since:**  
MHP1.0

# org.dvb.application

# AppProxy

## Syntax

```
public interface AppProxy
```

## All Known Subinterfaces:

[DVBHTMLProxy](#), [DVBJProxy](#)

## Description

An `AppProxy` Object is a proxy to an application. A call to the `start`, `stop` or `pause` will cause the resident Application Manager to respectively start, stop or pause the application bound to this `AppProxy` object. Each of these three method calls can throw a Security Exception if the calling application is not entitled to do so.

Each of these method call is asynchronous and will result in exactly one `AppStateChangeEvent` to be generated whether the method call was successful or not. If the method call was not successful, any call to the `hasFailed` method of the corresponding `AppStateChangeEvent` will return true.

Some of the methods here allow the `AppProxy` to transition through several states before the final state is reached. If this compound state transition is unsuccessful at any point, the resulting `AppStateChangeEvent` shall have a `fromstate` and a `tostate` which are both the last state in this transition which the `AppProxy` successfully entered. For instance, if an application were to call `start` on an `AppProxy` for a DVB-J application in the `NOT_LOADED` state and that DVB-J application was to throw a `XletStateChangeException` from its `startXlet` method, both the `getFromState` and `getToState` will return `PAUSED`.

Calling the `start` method for an application which is already running shall fail and generate an `AppStateChangeEvent` with `hasFailed` returning true and both `fromstate` and `tostate` being `STARTED`.

See the definition of `AppStateChangeEvent` for more information.

## See Also:

[AppStateChangeEvent](#)

## Fields

---

### DESTROYED

```
public static final int DESTROYED
```

The application is in the destroyed state.

---

### NOT\_LOADED

```
public static final int NOT_LOADED
```

The application has not yet been loaded from the network at all.

---

**PAUSED**

```
public static final int PAUSED
```

The application is in the paused state.

---

**STARTED**

```
public static final int STARTED
```

The application is in the active state.

## Methods

---

**addAppStateChangeListener(AppStateChangeListener)**

```
public void addAppStateChangeListener(AppStateChangeListener listener)
```

Add a listener to the application proxy so that an application can be informed if the application changes state.

**Parameters:**

`listener` - the listener to be added.

**Since:**

MHP1.0

---

**getState()**

```
public int getState()
```

Return the current state of the application.

**Returns:**

the state of the application.

---

**pause()**

```
public void pause()
```

Request that the application manager pause the application bound to this information structure.

The application will be paused. Calls to this method shall fail if the application is not in the active state. If the application represented by this AppProxy is a DVB-J application, calling this method will, if successful, result in the `pauseXlet` method being called on the Xlet making up the DVB-J application.

**Throws:**

`SecurityException` - if the application is not entitled to pause this application. Note that if an application is entitled to stop an application, it is also entitled to pause it: having the right to stop an application is logically equivalent to having the right to pause it.

**Since:**

MHP1.0

---

**removeAppStateChangeListener(AppStateChangeListener)**

```
public void removeAppStateChangeListener(AppStateChangeListener listener)
```

Remove a listener on the database.

**Parameters:**

`listener` - the listener to be removed.

**Since:**

MHP1.0

---

**resume()**

```
public void resume()
```

Request that the application manager resume the execution of the application. The `application` will be started. This method will throw a security exception if the application does not have the authority to resume the application. Calls to this method shall fail if the application is not in the paused state.

This method is asynchronous and its completion will be notified by an `AppStateChangeEvent`. In case of failure, the `hasFailed` method of the `AppStateChangeEvent` will return true. If the application represented by this `AppProxy` is a DVB-J application, calling this method will, if successful, result in the `startXlet` method being called on the Xlet making up the DVB-J application.

**Throws:**

`SecurityException` - if the application is not entitled to resume this application.

**Since:**

MHP1.0

---

**start()**

```
public void start()
```

Request that the application manager start the application bound to this information structure.

The `application` will be started. This method will throw a security exception if the application does not have the authority to start applications. Calls to this method shall only succeed if the application is in the not loaded or paused states. If the application was not loaded at the moment of this call, then the application will be started. In the case of a DVB-J application, it will be initialized and then started by the Application Manager, hence causing the Xlet to go from NotLoaded to Paused and then from Paused to Active. If the application was in the Paused state at the moment of the call and had never been in the Active state, then the application will be started. If the application represented by this `AppProxy` is a DVB-J application, calling this method will, if successful, result in the `startXlet` method being called on the Xlet making up the DVB-J application.

This method is asynchronous and its completion will be notified by an `AppStateChangeEvent`. In case of failure, the `hasFailed` method of the `AppStateChangeEvent` will return true.

**Throws:**

`SecurityException` - if the application is not entitled to start this application.

**Since:**

MHP1.0

---

**start(String[])**

```
public void start(java.lang.String[] args)
```

Request that the application manager start the application bound to this information structure passing to that application the specified parameters.

The `application` will be started. This method will throw a security exception if the application does not have the authority to start applications. Calls to this method shall only succeed if the application is in the not loaded or paused states. If the application was not loaded at the moment of this call, then the application will be started. In the case of a DVB-J application, it will be initialized and then started by the Application Manager, hence causing the Xlet to go from NotLoaded to Paused and then from Paused to Active. If the application was in the Paused state at the moment of the call and had never been in the Active state, then the application will be started. If the application represented by this `AppProxy` is a DVB-J application, calling this method will, if successful, result in the `startXlet` method being called on the Xlet making up the DVB-J application.

This method is asynchronous and its completion will be notified by an `AppStateChangeEvent`. In case of failure, the `hasFailed` method of the `AppStateChangeEvent` will return true.

**Parameters:**

`args` - the parameters to be passed into the application being started

**Throws:**

`SecurityException` - if the application is not entitled to start this application.

**Since:**

MHP1.0.1

---

**stop(boolean)**

```
public void stop(boolean forced)
```

Request that the application manager stop the application bound to this information structure.

The `application` will be stopped. A call to this method shall fail if the application was already in the destroyed state. This method call will stop the application if it was in any other state before the call. If the application is in the `NOT_LOADED` state then it shall move directly to the `DESTROYED` state with no other action being taken. If the application represented by this `AppProxy` is a DVB-J application and is not in the `DESTROYED` state then calling this method will, if successful, result in the `destroyXlet` method being called on the Xlet making up the DVB-J application with the same value for the parameter as passed to this method.

This method is asynchronous and its completion will be notified by an `AppStateChangeEvent`. In case of failure, the `hasFailed` method of the `AppStateChangeEvent` will return true.

**Parameters:**

`forced` - if true then do not ask the application but forcibly terminate it, if false give the application an opportunity to refuse.

**Throws:**

`SecurityException` - if the application is not entitled to stop this application.

**Since:**

MHP1.0



# org.dvb.application AppsControlPermission

## Syntax

```
public final class AppsControlPermission extends java.security.BasicPermission
```

```
java.lang.Object
|
+--java.security.Permission
|
+--java.security.BasicPermission
|
+--org.dvb.application.AppsControlPermission
```

## All Implemented Interfaces:

java.security.Guard, java.io.Serializable

## Description

This class represents a Permission to control the lifecycle of another application.

## Constructors

---

### AppsControlPermission()

```
public AppsControlPermission()
```

Creates a new AppsControlPermission. There is a simple mapping between the Application control Permission requests and the way the AppsControlPermission are granted. This mapping is defined in the main body of this specification.

---

### AppsControlPermission(String, String)

```
public AppsControlPermission(java.lang.String name, java.lang.String actions)
```

Creates a new AppsControlPermission. There is a simple mapping between the Application control Permission requests and the way the AppsControlPermission are granted. This mapping is defined in the main body of this specification. The actions string is currently unused and should be null. The name string is currently unused and should be empty. This constructor exists for use by the java.security.Policy object to instantiate new permission objects.

#### Parameters:

name - the name of the permission

actions - the actions string

## Methods

---

### equals(Object)

```
public boolean equals(java.lang.Object obj)
```

Checks for equality against this AppsControlPermission object.

**Overrides:**

java.security.BasicPermission.equals(java.lang.Object) in class java.security.BasicPermission

**Parameters:**

obj - the object to test for equality with this AppsControlPermission object.

**Returns:**

true if obj is an AppsControlPermission

---

**getActions()**

```
public java.lang.String getActions()
```

Returns the list of actions that had been passed to the constructor - it shall return null.

**Overrides:**

java.security.BasicPermission.getActions() in class java.security.BasicPermission

**Returns:**

a null String.

---

**hashCode()**

```
public int hashCode()
```

Returns the hash code value for this object.

**Overrides:**

java.security.BasicPermission.hashCode() in class java.security.BasicPermission

**Returns:**

the hash code value for this object.

---

**implies(Permission)**

```
public boolean implies(java.security.Permission permission)
```

Checks if this AppsControlPermission object "implies" the specified permission.

**Overrides:**

java.security.BasicPermission.implies(java.security.Permission) in class  
java.security.BasicPermission

**Parameters:**

permission - the specified permission to check.

**Returns:**

true if and only if the specified permission is an instance of AppsControlPermission, false otherwise.

# org.dvb.application AppsDatabase

## Syntax

```
public class AppsDatabase
    java.lang.Object
    |
    +--org.dvb.application.AppsDatabase
```

## Description

The `AppsDatabase` is an abstract view of the currently available applications. The entries will be provided by the application manager, and gleaned from the AIT signaling. When the service context in which an application is running undergoes service selection, instances of `AppsDatabase` used by that application shall be updated from the new service before an `AppsDatabaseEvent` is sent to the `newDatabase` method of any registered `AppsDatabaseEventListeners`.

Externally authorized applications shall not appear unless an instance of that application is actually running.

A generic launcher may be written which uses the database to display information in `AppAttributes` and uses an `AppProxy` to launch it

Methods on classes in this package do not block, they return the information the system currently has. Therefore applications should be aware that data may be stale, to within one refresh period of the AIT.

eg:

```
AppsDatabase theDatabase = AppsDatabase.getDatabase();
if (theDatabase != null) {
    Enumeration attributes = theDatabase.getAppAttributes();
    if(attributes != null) {
        while(attributes.hasMoreElements()) {
            AppAttributes info ;
            AppProxy proxy ;

            info = (AppAttributes)attributes.nextElement();
            proxy = (AppProxy)theDatabase.getAppProxy(info.getIdentifier());
            URL icon = info.getIcon();
            // blah blah..
            // lets start it.
            proxy.start(false, null);
        }
    }
}
```

Where methods on this class as specified as working on "available" applications or "currently available" applications the following definition shall apply. An application is "currently available" if and only if one of the following applies in the service context within which the application calling the method is executing.

- it is signalled as being present or autostart in the currently selected service of that service context and could be started.
- it is currently running in that service context.

In addition to the methods listed below, all calls made using an `AppsDatabaseFilter` shall only use that filter to test "currently available" applications as defined here.

Applications whose information (e.g. signaling) is invalid (e.g. one or more mandatory descriptors are missing or incorrect) may not be listed in the `AppsDatabase`. Where applications are signalled in a broadcast AIT and the MHP terminal tunes away from the service on which the AIT is carried, but without selecting a new service, the `AppsDatabase` shall retain the entries as signalled in that AIT until a new service is selected.

## Methods

---

### **addListener(AppsDatabaseEventListener)**

```
public void addListener(AppsDatabaseEventListener listener)
```

Add a listener to the database so that an application can be informed if the database changes.

**Parameters:**

`listener` - the listener to be added.

**Since:**

MHP1.0

---

### **getAppAttributes(AppID)**

```
public AppAttributes getAppAttributes(AppID key)
```

Returns the properties associated with the given ID. Returns null if no such application is available, or if the application is externally authorized.

Only one AppAttributes object shall be returned in the case where there are several applications having the same (organisationId, applicationId) pair. In such a case, the same algorithm as would be used to autostart such applications shall be used to decide between the available choices by the implementation.

This method shall return instances which reflect the contents of the database at the time the method is called. After an AppsDatabaseEvent has been generated, new instances may be returned. After a service selection has taken place, applications which survived the service selection may call this method in order to discover the attributes of the applications signalled on the new service.

**Parameters:**

`key` - an application ID.

**Returns:**

the value to which the key is mapped in this dictionary or `null` if the key is not an application ID, or not mapped to any application currently available.

**Since:**

MHP1.0

---

### **getAppAttributes(AppsDatabaseFilter)**

```
public java.util.Enumeration getAppAttributes(AppsDatabaseFilter filter)
```

Returns an enumeration of AppAttributes of the applications available. The Enumeration will contain the set of AppAttributes that satisfy the filtering criteria. For implementations conforming to this version of the specification, only `CurrentServiceFilter` filters may return a non empty Enumeration. If the filter object is not an instance of `CurrentServiceFilter` or a subclasses then, the method shall return an empty Enumeration.

This method shall return instances which reflect the contents of the database at the time the method is called. After an AppsDatabaseEvent has been generated, new instances may be returned. After a service selection has taken place, applications which survived the service selection may call this method in order to discover the attributes of the applications signalled on the new service.

No AppAttribute shall be returned for externally authorized applications, even ones which are executing. This method will return an empty Enumeration if there are no attributes.

**Parameters:**

`filter` - the filter to apply

**Returns:**

an enumeration of the applications attributes.

**Since:**

MHP1.0

---

**getAppIDs(AppsDatabaseFilter)**

```
public java.util.Enumeration getAppIDs(AppsDatabaseFilter filter)
```

Returns an enumeration of the application ID's available. The Enumeration will contain the set of AppID that match the filtering criteria. For implementations conforming to this version of the specification, only `CurrentServiceFilter` filters may return a non empty Enumeration. If the filter object is not an instance of `CurrentServiceFilter` or one of its subclasses then, the method shall return an empty Enumeration. No IDs shall be returned for externally authorized applications, even ones which are executing. This method will return an empty Enumeration if there are no matching applications.

**Parameters:**

`filter` - the filter to apply

**Returns:**

the applications available matching the filtering criteria

**Since:**

MHP1.0

---

**getAppProxy(AppID)**

```
public AppProxy getAppProxy(AppID key)
```

Returns the ApplicationProxy associated with the given ID. Returns null if no such application available.

Only one AppProxy object shall be returned in the case where there are several applications having the same (organisationId, applicationId) pair. In such a case, the same algorithm as would be used to autostart such applications shall be used to decide between the available choices by the implementation.

**Parameters:**

`key` - an application ID. `null` if the key is not an application ID, or not mapped to any application available.

**Returns:**

the value to which the key is mapped in this dictionary;

**Throws:**

`SecurityException` - if the calling application does not have the right to control the application associated with the given ID as defined by the security policy of the platform

**Since:**

MHP1.0

---

**getAppsDatabase()**

```
public static AppsDatabase getAppsDatabase()
```

Returns the singleton system-wide AppsDatabase object.

**Returns:**

the singleton AppsDatabase object.

**Since:**

MHP1.0

---

**removeListener(AppsDatabaseEventListener)**

```
public void removeListener(AppsDatabaseEventListener listener)
```

remove a listener on the database.

**Parameters:**

`listener` - the listener to be removed.

**Since:**

MHP1.0

---

**size()**

```
public int size()
```

Returns the number of applications currently available.

**Returns:**

the number of applications currently available.

**Since:**

MHP1.0

# org.dvb.application AppsDatabaseEvent

## Syntax

```
public class AppsDatabaseEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.dvb.application.AppsDatabaseEvent
```

## All Implemented Interfaces:

```
java.io.Serializable
```

## Description

The `AppsDatabaseEvent` class indicates either the an entry in the application database has changed, or so many changes have occurred. that the database should be considered totally new This event shall always be sent after switching to a new service. It is platform dependant if and when a new database event is thrown while tuned to the same service.

## Since:

```
MHP1.0
```

## Fields

---

### APP\_ADDED

```
public static final int APP_ADDED
```

The addition event id. The `APP_ADDED` event is generated whenever an entry is added to the `AppsDatabase`. It is NOT generated when the entry already in the `AppsDatabase` changes.

---

### APP\_CHANGED

```
public static final int APP_CHANGED
```

The changed event id. The `APP_CHANGED` event is generated whenever any of the information about an application changes. It is NOT generated when the entry is added to or removed from the `AppsDatabase`. In such cases, the `APP_ADDED` or `APP_DELETED` events will be generated instead.

---

### APP\_DELETED

```
public static final int APP_DELETED
```

The deletion event id. The `APP_DELETED` event is generated whenever an entry is removed from the `AppsDatabase`.

---

## NEW\_DATABASE

```
public static final int NEW_DATABASE
```

The new database event id.

## Constructors

---

### AppsDatabaseEvent(int, AppID, Object)

```
public AppsDatabaseEvent(int id, AppID appid, java.lang.Object source)
```

Create a new AppsDatabaseEvent object for the entry in the database that changed, or for a new database.

**Parameters:**

`id` - the cause of the event

`appid` - the AppID of the entry that changed

`source` - the AppsDatabase object.

**Since:**

MHP1.0

## Methods

---

### getAppID()

```
public AppID getAppID()
```

gets the application ID object for the entry in the database that changed.

When the event type is NEW\_DATABASE, AppID will be null.

**Returns:**

application ID representing the application

**Since:**

MHP1.0

---

### getEventId()

```
public int getEventId()
```

gets the type of the event.

**Returns:**

an integer that matches one of the static fields describing events.

**Since:**

MHP1.0



org.dvb.application

# AppsDatabaseEventListener

## Syntax

```
public interface AppsDatabaseEventListener extends java.util.EventListener
```

## All Superinterfaces:

java.util.EventListener

## Description

The `AppsDatabaseListener` class allows an application to monitor the application database so that it can keep an up to date interface without polling the state. The application shall receive these events in a timely fashion after the AIT changes, however it is system dependant how often the AIT table is checked.

## Since:

MHP1.0

## Methods

---

### entryAdded(AppsDatabaseEvent)

```
public void entryAdded(AppsDatabaseEvent evt)
```

The AppsDataBase has had an application entry added.

#### Parameters:

`evt` - the AppsDatabaseEvent.

#### Since:

MHP1.0

---

### entryChanged(AppsDatabaseEvent)

```
public void entryChanged(AppsDatabaseEvent evt)
```

The AppsDataBase has had an application entry changed.

#### Parameters:

`evt` - the AppsDatabaseEvent.

#### Since:

MHP1.0

---

### entryRemoved(AppsDatabaseEvent)

```
public void entryRemoved(AppsDatabaseEvent evt)
```

The AppsDataBase has had an application entry removed.

#### Parameters:

`evt` - the AppsDatabaseEvent.

#### Since:

MHP1.0

---

**newDatabase(AppsDatabaseEvent)**

```
public void newDatabase(AppsDatabaseEvent evt)
```

The AppsDataBase has radically changed.

**Parameters:**

`evt` - the AppsDatabaseEvent.

**Since:**

MHP1.0

# org.dvb.application AppsDatabaseFilter

## Syntax

```
public abstract class AppsDatabaseFilter
    java.lang.Object
    |
    +--org.dvb.application.AppsDatabaseFilter
```

## Direct Known Subclasses:

`CurrentServiceFilter`

## Description

Abstract class for the filters. Instances of concrete classes that extend `AppsDatabaseFilter` are passed to the `AppsDatabase.getAppAttributes` and `AppsDatabase.getAppIDs` methods to allow an applications to set a filter on the list of applications (respectively `AppAttributes` and `AppIDs`) that it wants to retrieve from the `AppDatabase`.

For this version of the specification, only one subclass is defined: `CurrentServiceFilter`

## Since:

MHP 1.0

## Constructors

---

### `AppsDatabaseFilter()`

```
public AppsDatabaseFilter()
```

Construct an `AppsDatabaseFilter` object.

## Methods

---

### `accept(AppID)`

```
public abstract boolean accept(AppID appid)
```

Test if a specified `appid` should be included in the Enumeration.

#### Parameters:

`appid` - the specified `appid` to test.

#### Returns:

true if the application with identifier `appid` should be listed, false otherwise.

#### Since:

MHP1.0

# org.dvb.application AppStateChangeEvent

## Syntax

```
public class AppStateChangeEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.dvb.application.AppStateChangeEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

The `AppStateChangeEvent` class indicates a state transition of the application. If the state transition was requested by an application through this API, the method `hasFailed` indicates whether the state change failed or not. Where a state change succeeds, `fromState` and `toState` shall indicate the original and destination state of the transition. If it failed, `fromState` shall return the state the application was in before the state transition was requested and the `toState` method shall return the state the application would have been in if the state transition had succeeded.

Attempting to start an application which is already running shall fail and generate an `AppStateChangeEvent` with `hasFailed` returning true and both `fromstate` and `tostate` being `STARTED`.

## Since:

MHP1.0

## Constructors

---

### AppStateChangeEvent(AppID, int, int, Object, boolean)

```
public AppStateChangeEvent(AppID appid, int fromstate, int tostate,
    java.lang.Object source, boolean hasFailed)
```

Create an `AppStateChangeEvent` object.

#### Parameters:

`appid` - a registry entry representing the tracked application

`fromstate` - the state the application was in before the state transition was requested, where the value of `fromState` is one of the state values defined in the `AppProxy` interface or in the interfaces inheriting from it

`tostate` - state the application would be in if the state transition succeeds, where the value of `toState` is one of the state values defined in the `AppProxy` interface or in the interfaces inheriting from it

`hasFailed` - an indication of whether the transition failed (true) or succeeded (false)

`source` - the `AppProxy` where the state transition happened

## Methods

---

### getAppID()

```
public AppID getAppID()
```

The application the listener was tracking has made a state transition from `fromState` to `toState`.

**Returns:**

a registry entry representing the tracked application

**Since:**

MHP1.0

---

### getFromState()

```
public int getFromState()
```

The application the listener is tracking was in `fromState`, where the value of `fromState` is one of the state values defined in the `AppProxy` interface or in the interfaces inheriting from it.

**Returns:**

the old state

**Since:**

MHP1.0

---

### getToState()

```
public int getToState()
```

The application the listener is tracking is now in `toState`, where the value of `toState` is one of the state values defined in the `AppProxy` interface or in the interfaces inheriting from it.

**Returns:**

the intended new state

**Since:**

MHP1.0

---

### hasFailed()

```
public boolean hasFailed()
```

This method determines whether an attempt to change the state of an application has failed.

**Returns:**

true if the attempt to change the state of the application failed, false otherwise

**Since:**

MHP1.0

org.dvb.application

# AppStateChangeListener

## Syntax

```
public interface AppStateChangeListener extends java.util.EventListener
```

## All Superinterfaces:

java.util.EventListener

## Description

The `AppStateChangeListener` class allows a launcher application to keep track of applications it launches or other applications running as part of the same service.

## Since:

MHP1.0

## Methods

---

### `stateChange(AppStateChangeEvent)`

```
public void stateChange(AppStateChangeEvent evt)
```

The application the listener was tracking has made a state transition from `fromState` to `toState` and this method will be given the state event.

#### Parameters:

`evt` - the `AppStateChangeEvent`.

#### Since:

MHP1.0

# org.dvb.application CurrentServiceFilter

## Syntax

```
public class CurrentServiceFilter extends AppsDatabaseFilter
```

```
java.lang.Object
|
+--AppsDatabaseFilter
|
+--org.dvb.application.CurrentServiceFilter
```

## Description

Instances of `CurrentServiceFilter` are used to set a filter on the list of applications that are retrieved from the `AppsDatabase` (See methods `getAppsAttributes` and `getAppIDs`)

For this version of the specification, only the `CurrentServiceFilter` class is defined. A `CurrentServiceFilter` is used to indicate that only broadcast applications that are signalled in one of the AITs of the current service shall be returned by the `getAppsAttributes` and `getAppIDs` methods of `AppsDatabase`. Subclasses of `CurrentServiceFilter` can override the `accept` method so as to implement their own filter criteria on the `AppID`'s values.

## Since:

MHP 1.0

## Constructors

---

### CurrentServiceFilter()

```
public CurrentServiceFilter()
public Constructor of the CurrentServiceFilter
```

## Methods

---

### accept(AppID)

```
public boolean accept(AppID appid)
```

Test if a specified `appid` should be included in the Enumeration.

#### Overrides:

`accept(AppID)` in class `AppsDatabaseFilter`

#### Parameters:

`appid` - the specified `appid` to test.

#### Returns:

true if the application with identifier `appid` should be listed, false otherwise.

#### Since:

MHP1.0

# org.dvb.application DVBHTMLProxy

## Syntax

```
public interface DVBHTMLProxy extends AppProxy
```

## All Superinterfaces:

[AppProxy](#)

## Description

A `DVBHTMLProxy` Object is a proxy to a DVBHTML application.

## Methods

---

### **prefetch()**

```
public void prefetch()
```

Loads the initial entry page of the application and waits for a signal. This method mimics the PREFETCH control code and is intended to be called instead of and not as well as start. Calling prefetch on a started application will have no effect.

#### **Throws:**

`SecurityException` - if the calling application does not have permission to start applications

#### **Since:**

MHP1.0

---

### **startTrigger(Date)**

```
public void startTrigger(java.util.Date starttime)
```

Sends the application a start trigger at the specified time.

#### **Parameters:**

`starttime` - the specified time to send a start trigger to the application. If the time has already passed the application manager shall send the trigger immediately. Dates pre-epoch shall always cause the application manager to send the trigger immediately.

#### **Throws:**

`SecurityException` - if the calling application does not have permission to start applications

#### **Since:**

MHP1.0

---

### **trigger(Date, Object)**

```
public void trigger(java.util.Date time, java.lang.Object triggerPayload)
```

Sends the application a trigger with the given payload at the specified time.



**Parameters:**

`time` - the specified time to send a start trigger to the application. If the time has already passed the application manager should send the trigger immediately. Dates pre-epoch shall always cause the application manager to send a 'now' trigger.

`triggerPayload` - the specified payload to deliver with the trigger. The payload is specified as object, but this will be refined once DVB-HTML Triggers are properly defined.

**Throws:**

`SecurityException` - if the calling application does not have permission to start applications

**Since:**

MHP1.0

# org.dvb.application DVBJProxy

## Syntax

```
public interface DVBJProxy extends AppProxy
```

## All Superinterfaces:

[AppProxy](#)

## Description

A `DVBJProxy` Object is a proxy to a DVBJ application.

## Fields

---

### LOADED

```
public static final int LOADED
```

The application is in the loaded state.

## Methods

---

### init()

```
public void init()
```

Requests the application manager calls the `initXlet` method on the application.

This method is asynchronous and its completion will be notified by an `AppStateChangeEvent`. In case of failure, the `hasFailed` method of the `AppStateChangeEvent` will return true. Calls to this method shall only succeed if the application is in the `NOT_LOADED` or `LOADED` states. If the application is in the `NOT_LOADED` state, the application will move through the `LOADED` state into the `PAUSED` state before calls to this method complete.

In all cases, an `AppStateChangeEvent` will be sent, whether the call was successful or not.

#### Throws:

`SecurityException` - if the application is not entitled to load this application. being able to load an application requires to be entitled to start it.

#### Since:

MHP1.0

---

### load()

```
public void load()
```

Provides a hint to preload at least the initial class of the application into local storage, resources permitting. This does not require loading of classes into the virtual machine or creation of a new logical virtual machine which are implications of the `init` method.

This method is asynchronous and its completion will be notified by an `AppStateChangedEvent`. In case of failure, the `hasFailed` method of the `AppStateChangedEvent` will return `true`. Calls to this method shall only succeed if the application is in the `NOT_LOADED` state. In all cases, an `AppStateChangeEvent` will be sent, whether the call was successful or not.

**Throws:**

`SecurityException` - if the application is not entitled to load this application. being able to load an application requires to be entitled to start it.

**Since:**

MHP1.0

org.dvb.application

# IllegalProfileParameterException

## Syntax

```
public class IllegalProfileParameterException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--org.dvb.application.IllegalProfileParameterException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

The `IllegalProfileParameter` exception is thrown if the application attempts to ask for a version number for a profile not specified for the application.

## Since:

MHP1.0

## Constructors

---

### IllegalProfileParameterException()

```
public IllegalProfileParameterException()
```

Construct a `IllegalProfileParameterException` with no detail message

---

### IllegalProfileParameterException(String)

```
public IllegalProfileParameterException(java.lang.String s)
```

Construct a `IllegalProfileParameterException` with a detail message

#### Parameters:

`s` - detail message

org.dvb.application

# LanguageNotAvailableException

## Syntax

```
public class LanguageNotAvailableException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
|
+--java.lang.Exception
|
+--org.dvb.application.LanguageNotAvailableException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

The `LanguageNotAvailableException` exception is thrown if the application asks for the name of an application in a language not signalled in the AIT.

## Since:

MHP1.0

## Constructors

---

### LanguageNotAvailableException()

```
public LanguageNotAvailableException()
```

Construct a `LanguageNotAvailableException` with no detail message

---

### LanguageNotAvailableException(String)

```
public LanguageNotAvailableException(java.lang.String s)
```

Construct a `LanguageNotAvailableException` with a detail message

#### Parameters:

`s` - detail message

---

## **Annex T (normative): Permissions**

# Package org.dvb.net.ca

## Description

Provides extensions to the conditional access API from DAVIC.

### Class Summary

#### Classes

[CAPermission](#)

This class is for CA permissions.

# org.dvb.net.ca CAPermission

## Syntax

```
public class CAPermission extends java.security.BasicPermission
```

```
java.lang.Object
|
+--java.security.Permission
|
+--java.security.BasicPermission
|
+--org.dvb.net.ca.CAPermission
```

## All Implemented Interfaces:

java.security.Guard, java.io.Serializable

## Description

This class is for CA permissions. A CAPermission contains a name, but no actions list.

A CAPermission contains a range of CA system ids and a specific permission for that range of CA system ids. Instead of a range of CA system ids, the CAPermission can also refer to a single CA system id.

The name has the following syntax:

CASystemIdRange ":" Permission

where CASystemIdRange = CASystemId [ "-" CASystemId ] | ""

and Permission = "MMI" | "buy" | "entitlementInfo" | "messagePassing" | ""

Examples:

- 0x1200-0x120A:buy (The permission to buy entitlement for all the CA systems with ids between 0x1200 and 0x120A inclusive.)
- 0x1201:entitlementInfo (The permission to get entitlement information for the CA system with id 0x1201)
- 0x120d:\* (This wildcard expresses all the permissions for the CA system with id 0x120d).

Note: The CASystemId is expressed as a hexadecimal value.

The permission "MMI" corresponds with the SecurityException on CAModuleManager.addMMIListener(). The permission "buy" corresponds with the SecurityException on CAModule.buyEntitlement(). The permission "entitlementInfo" corresponds with the SecurityException on CAModule.queryEntitlement() and CAModule.listEntitlements(). The permission "messagePassing" corresponds with CAModule.openMessageSession(MessageListener)

## Constructors

---

### CAPermission(String)

```
public CAPermission(java.lang.String name)
```

Creates a new CAPermission with the specified name. The name is the symbolic name of the CAPermission.

#### Parameters:

name - the name of the CAPermission



---

**CAPermission(String, String)**

```
public CAPermission(java.lang.String name, java.lang.String actions)
```

Creates a new CAPermission object with the specified name. The name is the symbolic name of the CAPermission, and the actions String is unused and should be null. This constructor exists for use by the Policy object to instantiate new Permission objects.

**Parameters:**

`name` - the name of the CAPermission

`actions` - should be null.

# Package org.dvb.net.tuning

## Description

Provides extensions to the tuning API from DAVIC.

### Class Summary

#### Classes

<code>DvbNetworkInterface-</code> <code>SIUtil</code>	Each SI database is associated with a network interface and vice versa.
<code>TunerPermission</code>	This class is for tuner permissions.

# org.dvb.net.tuning DvbNetworkInterfaceSIUtil

## Syntax

```
public class DvbNetworkInterfaceSIUtil  
  
java.lang.Object  
|  
+--org.dvb.net.tuning.DvbNetworkInterfaceSIUtil
```

## Description

Each SI database is associated with a network interface and vice versa. This class allows the application to query this association.

## Since:

MHP 1.0.1

## Methods

---

### getNetworkInterface(SIDatabase)

```
public static org.davic.net.tuning.NetworkInterface getNetworkInterface(SIDatabase sd)
```

Gets the network interface for a particular SI database.

#### Parameters:

`sd` - the SI database for which the associated network interface will be returned.

#### Returns:

the associated network interface

---

### getSIDatabase(NetworkInterface)

```
public static SIDatabase getSIDatabase(org.davic.net.tuning.NetworkInterface ni)
```

Gets the SI database for a particular network interface.

#### Parameters:

`ni` - the network interface for which the associated SI database will be returned.

#### Returns:

the associated SI database

# org.dvb.net.tuning TunerPermission

## Syntax

```
public class TunerPermission extends java.security.BasicPermission
```

```
java.lang.Object
|
+--java.security.Permission
|
+--java.security.BasicPermission
|
+--org.dvb.net.tuning.TunerPermission
```

## All Implemented Interfaces:

java.security.Guard, java.io.Serializable

## Description

This class is for tuner permissions. A TunerPermission contains no name and no actions list. If an application has the tuner permission, then it shall not receive a `SecurityException` from those methods in that API defined to throw one. Without such a permission, it shall receive such an exception.

## Constructors

---

### TunerPermission(String)

```
public TunerPermission(java.lang.String name)
```

Creates a new TunerPermission. The name string is currently unused and should be empty.

#### Parameters:

`name` - the name of the TunerPermission.

---

### TunerPermission(String, String)

```
public TunerPermission(java.lang.String name, java.lang.String actions)
```

Creates a new TunerPermission. The name string is currently unused and should be empty. The actions string is currently unused and should be null. This constructor exists for use by the Policy object to instantiate new Permission objects.

#### Parameters:

`name` - the name of the TunerPermission.

`actions` - the actions list

---

## **Annex U (normative): Extended graphics APIs**

# Package org.dvb.ui

## Description

Provides extended graphics functionality.

Class Summary	
<b>Interfaces</b>	
<code>TestOpacity</code>	Interface implemented by Components or Containers in order to allow the platform to query whether their paint method is fully opaque.
<code>TextOverflowListener</code>	The <code>TextOverflowListener</code> is an interface that an application may implement and register in the <code>DVBTextLayoutManager</code> .
<b>Classes</b>	
<code>DVBAlphaComposite</code>	This <code>DVBAlphaComposite</code> class implements the basic alpha compositing rules for combining source and destination pixels to achieve blending and transparency effects with graphics, images and video.
<code>DVBBufferedImage</code>	The <code>DVBBufferedImage</code> subclass describes an <code>java.awt.Image</code> with an accessible buffer of image data.
<code>DVBColor</code>	A <code>Color</code> class which adds the notion of alpha.
<code>DVBGraphics</code>	The <code>DVBGraphics</code> class is a adapter class to support alpha compositing in an MHP device.
<code>DVBTextLayoutManager</code>	The <code>DVBTextLayoutManager</code> provides a text rendering layout mechanism for the <code>org.havi.ui.HStaticText</code> <code>org.havi.ui.HText</code> and <code>org.havi.ui.HTextButton</code> classes.
<code>FontFactory</code>	Provides a mechanism for applications to instantiate fonts that are not built into the system.
<b>Exceptions</b>	
<code>DVBRasterFormatException</code>	This exception is thrown for some invalid operations on instances of <code>DVBBufferedImage</code> .
<code>FontFormatException</code>	Thrown when attempt is made to read a file describing a font when the contents of that file are not valid.
<code>FontNotAvailableException</code>	Thrown when attempt is made to instantiate a font that cannot be located.
<code>UnsupportedDrawingOperationException</code>	The <code>UnsupportedDrawingOperationException</code> class represents an exception that is thrown if an drawing operation is not supported on this platform.

# org.dvb.ui DVBAAlphaComposite

## Syntax

```
public final class DVBAAlphaComposite
    java.lang.Object
    |
    +--org.dvb.ui.DVBAAlphaComposite
```

## Description

This `DVBAAlphaComposite` class implements the basic alpha compositing rules for combining source and destination pixels to achieve blending and transparency effects with graphics, images and video. The rules implemented by this class are a subset of the Porter-Duff rules described in T. Porter and T. Duff, "Compositing Digital Images", SIGGRAPH 84, 253-259.

If any input does not have an alpha channel, an alpha value of 1.0, which is completely opaque, is assumed for all pixels. A constant alpha value can also be specified to be multiplied with the alpha value of the source pixels.

The following abbreviations are used in the description of the rules:

- $C_s$  = one of the color components of the source pixel without alpha.
- $cs$  = color component of a source pixel premultiplied with alpha ( $cs = A_s * A_r * C_s$ )
- $C_d$  = one of the color components of the destination pixel without alpha.
- $cd$  = color component of a destination pixel premultiplied with alpha
- $C_n$  = the new constructed color without alpha.
- $cn$  = the new constructed color premultiplied with alpha
- $A_s$  = alpha component of the source pixel.
- $A_d$  = alpha component of the destination pixel.
- $A_n$  = the new alpha after compositing
- $A_r$  = alpha, specified by `getInstance(int Rule, float Ar)`. Unless otherwise specified  $A_r = 1.0$
- $F_s$  = fraction of the source pixel that contributes to the output.
- $F_d$  = fraction of the input destination pixel that contributes to the output.

The color and alpha components produced by the compositing operation are calculated as follows:

$$cn = (As * Ar) * Cs * Fs + Ad * Cd * Fd$$

$$An = (As * Ar) * Fs + Ad * Fd$$

$$Cn = cn / An$$

where  $F_s$  and  $F_d$  are specified by each rule.

The alpha resulting from the compositing operation is stored in the destination if the destination has an alpha channel. Otherwise, the resulting color is divided by the resulting alpha before being stored in the destination and the alpha is discarded. If the alpha value is 0.0, the color values are set to 0.0.

## See Also:

`java.awt.AlphaComposite`

## Fields

---

### Clear

```
public static final DVBAAlphaComposite Clear
```

DVBAlphaComposite object that implements the opaque CLEAR rule with an alpha (Ar) of 1.0f.

**See Also:**

[CLEAR](#)

## CLEAR

```
public static final int CLEAR
```

Porter-Duff Clear rule. Both the color and the alpha of the destination are cleared. Neither the source nor the destination is used as input.

$F_s = 0$  and  $F_d = 0$ , thus:

```
cn = 0
An = 0
Cn = 0
```

**Note that this operation is a fast drawing operation** This operation is the same as using a source with alpha= 0 and the SRC rule

## DST\_IN

```
public static final int DST_IN
```

Porter-Duff Destination In Source rule. The part of the destination lying inside of the source replaces the destination.

$F_s = 0$  and  $F_d = (A_s * A_r)$ , thus:

```
cn = Ad * Cd * (As * Ar)
An = Ad * (As * Ar)
Cn = Cd
```

**Note that this operation is faster than e.g. SRC\_OVER but slower as SRC**

## DST\_OUT

```
public static final int DST_OUT
```

Porter-Duff Destination Held Out By Source rule. The part of the destination lying outside of the source replaces the destination.

$F_s = 0$  and  $F_d = (1 - (A_s * A_r))$ , thus:

```
cn = Ad * Cd * (1 - (As * Ar))
An = Ad * (1 - (As * Ar))
Cn = Cd
```

**Note that this operation is faster than e.g. SRC\_OVER but slower as SRC**

## DST\_OVER

```
public static final int DST_OVER
```

Porter-Duff Destination Over Source rule. The destination is composited over the source and the result replaces the destination.

$F_s = (1 - A_d)$  and  $F_d = 1$ , thus:

```
cn = (As * Ar) * Cs * (1 - Ad) + Ad * Cd
An = (As * Ar) * (1 - Ad) + Ad
```

**Note that this can be a very slow drawing operation**

## DstIn

```
public static final DVBAAlphaComposite DstIn
```



DVBAlphaComposite object that implements the opaque DST\_IN rule with an alpha (Ar) of 1.0f.

**See Also:**

DST\_IN

### DstOut

```
public static final DVBAlphaComposite DstOut
```

DVBAlphaComposite object that implements the opaque DST\_OUT rule with an alpha (Ar) of 1.0f.

**See Also:**

DST\_OUT

### DstOver

```
public static final DVBAlphaComposite DstOver
```

DVBAlphaComposite object that implements the opaque DST\_OVER rule with an alpha (Ar) of 1.0f.

**See Also:**

DST\_OVER

### Src

```
public static final DVBAlphaComposite Src
```

DVBAlphaComposite object that implements the opaque SRC rule with an alpha (Ar) of 1.0f.

**See Also:**

SRC

### SRC

```
public static final int SRC
```

Porter-Duff Source rule. The source is copied to the destination. The destination is not used as input.

Fs = 1 and Fd = 0, thus:

```
cn = (As*Ar)*Cs
An = As*Ar
Cn = Cs
```

**Note that this is a fast drawing routine**

### SRC\_IN

```
public static final int SRC_IN
```

Porter-Duff Source In Destination rule. The part of the source lying inside of the destination replaces the destination.

Fs = Ad and Fd = 0, thus:

```
cn = (As*Ar)*Cs*Ad
An = (As*Ar)*Ad
Cn = Cs
```

**Note that this operation is faster than e.g. SRC\_OVER but slower as SRC**

### SRC\_OUT

```
public static final int SRC_OUT
```

Porter-Duff Source Held Out By Destination rule. The part of the source lying outside of the destination replaces the destination.

$F_s = (1 - A_d)$  and  $F_d = 0$ , thus:

```
cn = (As*Ar)*Cs*(1-Ad)
An = (As*Ar)*(1-Ad)
Cn = Cs
```

**Note that this operation is faster than e.g. SRC\_OVER but slower as SRC**

## SRC\_OVER

```
public static final int SRC_OVER
```

Porter-Duff Source Over Destination rule. The source is composited over the destination.

$F_s = 1$  and  $F_d = (1 - (A_s * A_r))$ , thus:

```
cn = (As*Ar)*Cs + Ad*Cd*(1-(As*Ar))
An = (As*Ar) + Ad*(1-(As*Ar))
```

**Note that this can be a very slow drawing operation**

## SrcIn

```
public static final DVBAAlphaComposite SrcIn
```

DVBAAlphaComposite object that implements the opaque SRC\_IN rule with an alpha ( $A_r$ ) of 1.0f.

**See Also:**

[SRC\\_IN](#)

## SrcOut

```
public static final DVBAAlphaComposite SrcOut
```

DVBAAlphaComposite object that implements the opaque SRC\_OUT rule with an alpha ( $A_r$ ) of 1.0f.

**See Also:**

[SRC\\_OUT](#)

## SrcOver

```
public static final DVBAAlphaComposite SrcOver
```

DVBAAlphaComposite object that implements the opaque SRC\_OVER rule with an alpha ( $A_r$ ) of 1.0f.

**See Also:**

[SRC\\_OVER](#)

## Methods

### equals(Object)

```
public boolean equals(java.lang.Object obj)
```

Tests if the specified `java.lang.Object` is equal to this DVBAAlphaComposite object.

**Overrides:**

`java.lang.Object.equals(java.lang.Object)` in class `java.lang.Object`

**Parameters:**

obj - the Object to test for equality

**Returns:**

true if obj is a DVBAAlphaComposite and has the same values for rule and alpha as this object. Otherwise false shall be returned.

---

**getAlpha()**

```
public float getAlpha()
```

Returns the alpha value of this DVBAAlphaComposite. If this DVBAAlphaComposite does not have an alpha value, 1.0 is returned.

**Returns:**

the alpha value of this DVBAAlphaComposite.

---

**getInstance(int)**

```
public static DVBAAlphaComposite getInstance(int rule)
```

Creates an DVBAAlphaComposite object with the specified rule. The value for alpha shall be 1.0f.

**Parameters:**

rule - the compositing rule

**Returns:**

an DVBAAlphaComposite object with the specified rule.

---

**getInstance(int, float)**

```
public static DVBAAlphaComposite getInstance(int rule, float alpha)
```

Creates an DVBAAlphaComposite object with the specified rule and the constant alpha (Ar) to multiply with the alpha of the source (As). The source is multiplied with the specified alpha before being composited with the destination.

**Parameters:**

rule - the compositing rule

alpha - the constant alpha (Ar) to be multiplied with the alpha of the source (As). alpha must be a floating point number in the inclusive range [0.0, 1.0].

**Returns:**

an DVBAAlphaComposite object with the specified rule and the constant alpha to multiply with the alpha of the source.

---

**getRule()**

```
public int getRule()
```

Returns the compositing rule of this DVBAAlphaComposite.

**Returns:**

the compositing rule of this DVBAAlphaComposite.

# org.dvb.ui DVBBufferedImage

## Syntax

```
public class DVBBufferedImage extends java.awt.Image
```

```
java.lang.Object
|
+--java.awt.Image
|
+--org.dvb.ui.DVBBufferedImage
```

## Description

The `DVBBufferedImage` subclass describes an `java.awt.Image` with an accessible buffer of image data. The `DVBBufferedImage` is an adapter class for `java.awt.image.BufferedImage`. It supports two different platform dependent sample models `TYPE_BASE` and `TYPE_ADVANCED`. Buffered images with the `TYPE_BASE` have the same sample model as the on screen graphics buffer, thus `TYPE_BASE` could be CLUT based. `TYPE_ADVANCED` has a direct color model but it is not specified how many bits are used to store the different color components. By default, a new `DVBBufferedImage` is transparent. All alpha values are set to 0;

## Since:

MHP 1.0

## Fields

---

### TYPE\_ADVANCED

```
public static final int TYPE_ADVANCED
```

Represents an image stored in a best possible `SampleModel` (platform dependent) The image has a `DirectColorModel` with alpha. The color data in this image is considered not to be premultiplied with alpha. The data returned by `getRGB()` will be in the `TYPE_INT_ARGB` color model that is alpha component in bits 24-31, the red component in bits 16-23, the green component in bits 8-15, and the blue component in bits 0-7. The data for `setRGB()` shall be in the `TYPE_INT_ARGB` color model as well.

#### Since:

MHP 1.0

---

### TYPE\_BASE

```
public static final int TYPE_BASE
```

Represents an image stored in a platform dependent `Sample Model`. This color model is not visible to applications. The data returned by `getRGB()` will be in the `TYPE_INT_ARGB` color model that is alpha component in bits 24-31, the red component in bits 16-23, the green component in bits 8-15, and the blue component in bits 0-7. The data for `setRGB()` shall be in the `TYPE_INT_ARGB` color model as well.

#### Since:

MHP 1.0

## Constructors

---

### DVBBufferedImage(int, int)

```
public DVBBufferedImage(int width, int height)
```

Constructs a DVBBufferedImage with the specified width and height. The Sample Model used the image is the native Sample Model (TYPE\_BASE) of the implementation. Note that a request can lead to an `java.lang.OutOfMemoryError`. Applications should be aware of this.

**Parameters:**

`width` - the width of the created image

`height` - the height of the created image

**Since:**

MHP 1.0

---

### DVBBufferedImage(int, int, int)

```
public DVBBufferedImage(int width, int height, int type)
```

Constructs a new DVBBufferedImage with the specified width and height in the Sample Model specified by type. Note that a request can lead to an `java.lang.OutOfMemoryError`. Applications should be aware of this.

**Parameters:**

`width` - the width of the DVBBufferedImage

`height` - the height of the DVBBufferedImage

`type` - the ColorSpace of the DVBBufferedImage

**Since:**

MHP 1.0

---

## Methods

---

### createGraphics()

```
public DVBGraphics createGraphics()
```

Creates a DVBGraphics, which can be used to draw into this DVBBufferedImage.

**Returns:**

a DVBGraphics, used for drawing into this image.

**Since:**

MHP 1.0

---

### dispose()

```
public void dispose()
```

Disposes of this buffered image. This method releases the resources (e.g. pixel memory) underlying this buffered image. After calling this method ;

- the image concerned may not be used again
- the image shall be considered to have a width and height of -1, -1 as specified for instances of `java.awt.Image` where the width and height are not yet known.

- the `getGraphics` method may return null

**Since:**

MHP 1.0.1

---

**flush()**

```
public void flush()
```

Flushes all resources being used to cache optimization information. The underlying pixel data is unaffected.

**Overrides:**

`java.awt.Image.flush()` in class `java.awt.Image`

---

**getGraphics()**

```
public java.awt.Graphics getGraphics()
```

This method returns a `java.awt.Graphics`, it is here for backwards compatibility. `createGraphics()` is more convenient, since it is declared to return a `DVBGraphics`.

**Overrides:**

`java.awt.Image.getGraphics()` in class `java.awt.Image`

**Returns:**

a `Graphics`, which can be used to draw into this image.

---

**getHeight()**

```
public int getHeight()
```

Returns the height of the `DVBBufferedImage`.

**Returns:**

the height of this `DVBBufferedImage`.

**Since:**

MHP 1.0

---

**getHeight(ImageObserver)**

```
public int getHeight(java.awt.image.ImageObserver observer)
```

Returns the actual height of the image. If the height is not known yet then the `ImageObserver` is notified later and `-1` is returned.

**Overrides:**

`java.awt.Image.getHeight(java.awt.image.ImageObserver)` in class `java.awt.Image`

**Parameters:**

`observer` - the `ImageObserver` that receives information about the image

**Returns:**

the height of the image or `-1` if the height is not yet known.

**See Also:**

`java.awt.Image.getWidth(ImageObserver)`, `java.awt.image.ImageObserver`

---

**getImage()**

```
public java.awt.Image getImage()
```

Returns a `java.awt.Image` representing this buffered image. In implementations using the JDK1.2 API this returns a `java.awt.image.BufferedImage` cast to a `java.awt.Image`. Otherwise it is implementation dependent whether it returns this image or whether it returns an instance of an underlying platform specific sub-class of `java.awt.Image`.

**Returns:**

a `java.awt.image` representing this buffered image

**Since:**

MHP 1.0

---

**getProperty(String, ImageObserver)**

```
public java.lang.Object getProperty(java.lang.String name,  
    java.awt.image.ImageObserver observer)
```

Returns a property of the image by name. Individual property names are defined by the various image formats. If a property is not defined for a particular image, this method returns the `UndefinedProperty` field. If the properties for this image are not yet known, then this method returns `null` and the `ImageObserver` object is notified later. The property name "comment" should be used to store an optional comment that can be presented to the user as a description of the image, its source, or its author.

**Overrides:**

`java.awt.Image.getProperty(java.lang.String, java.awt.image.ImageObserver)` in class `java.awt.Image`

**Parameters:**

`name` - the property name

`observer` - the `ImageObserver` that receives notification regarding image information

**Returns:**

an `java.lang.Object` that is the property referred to by the specified `name` or `null` if the properties of this image are not yet known.

**See Also:**

`java.awt.image.ImageObserver`, `java.awt.Image.UndefinedProperty`

---

**getRGB(int, int)**

```
public int getRGB(int x, int y)
```

Returns the specified integer pixel in the default RGB color model (`TYPE_INT_ARGB`) and default sRGB colorspace. Color conversion takes place if the used Sample Model is not 8-bit for each color component. There are only 8-bits of precision for each color component in the returned data when using this method. Note that when a lower precision is used in this buffered image `getRGB` may return different values than those used in `setRGB()`

**Parameters:**

`x` - the x-coordinate of the pixel

`y` - the y-coordinate of the pixel

**Returns:**

an integer pixel in the default RGB color model (`TYPE_INT_ARGB`) and default sRGB colorspace.

**Throws:**

`ArrayIndexOutOfBoundsException` - if `x` or `y` is out of bounds

**Since:**

MHP 1.0

---

**getRGB(int, int, int, int, int[], int, int)**

```
public int[] getRGB(int startX, int startY, int w, int h, int[] rgbArray, int offset,
                  int scansize)
```

Returns an array of integer pixels in the default RGB color model (TYPE\_INT\_ARGB) and default sRGB color space, from a rectangular region of the image data. There are only 8-bits of precision for each color component in the returned data when using this method. With a specified coordinate (x, y) in the image, the ARGB pixel can be accessed in this way:

```
pixel = rgbArray[offset + (y-startY)*scansize + (x-startX)];
```

**Parameters:**

`startX` - the x-coordinate of the upper-left corner of the specified rectangular region

`startY` - the y-coordinate of the upper-left corner of the specified rectangular region

`w` - the width of the specified rectangular region

`h` - the height of the specified rectangular region

`rgbArray` - if not null, the rgb pixels are written here

`offset` - offset into the `rgbArray`

`scansize` - scanline stride for the `rgbArray`

**Returns:**

array of ARGB pixels.

**Throws:**

`ArrayIndexOutOfBoundsException` - if the specified portion of the image data is out of bounds.

`IllegalArgumentException`

**Since:**

MHP 1.0

---

**getSource()**

```
public java.awt.image.ImageProducer getSource()
```

Returns the object that produces the pixels for the image.

**Overrides:**

`java.awt.Image.getSource()` in class `java.awt.Image`

**Returns:**

the `java.awt.image.ImageProducer` that is used to produce the pixels for this image.

**See Also:**

`java.awt.image.ImageProducer`

---

**getSubimage(int, int, int, int)**

```
public DVBufferedImage getSubimage(int x, int y, int w, int h)
```

Returns a subimage defined by a specified rectangular region. The returned `DVBufferedImage` shares the same data array as the original image.

**Parameters:**

`x` - the x-coordinate of the upper-left corner of the specified rectangular region

`y` - the y-coordinate of the upper-left corner of the specified rectangular region

`w` - the width of the specified rectangular region



h - the height of the specified rectangular region

**Returns:**

a `DVBBufferedImage` that is the subimage of this `DVBBufferedImage`.

**Throws:**

`DVBRasterFormatException` - if the specified area is not contained within this `DVBBufferedImage`.

**Since:**

MHP 1.0

**getWidth()**

```
public int getWidth()
```

Returns the width of the `DVBBufferedImage`.

**Returns:**

the width of this `DVBBufferedImage`.

**Since:**

MHP 1.0

**getWidth(ImageObserver)**

```
public int getWidth(java.awt.image.ImageObserver observer)
```

Returns the actual width of the image. If the width is not known yet then the `java.awt.image.ImageObserver` is notified later and `-1` is returned.

**Overrides:**

`java.awt.Image.getWidth(java.awt.image.ImageObserver)` in class `java.awt.Image`

**Parameters:**

`observer` - the `ImageObserver` that receives information about the image

**Returns:**

the width of the image or `-1` if the width is not yet known.

**See Also:**

`java.awt.Image.getHeight(ImageObserver)`, `java.awt.image.ImageObserver`

**setRGB(int, int, int)**

```
public synchronized void setRGB(int x, int y, int rgb)
```

Sets a pixel in this `DVBBufferedImage` to the specified ARGB value. The pixel is assumed to be in the default RGB color model, `TYPE_INT_ARGB`, and default sRGB color space.

**Parameters:**

x - the x-coordinate of the pixel to set

y - the y-coordinate of the pixel to set

rgb - the ARGB value

**Since:**

MHP 1.0

**setRGB(int, int, int, int, int[], int, int)**

```
public void setRGB(int startX, int startY, int w, int h, int[] rgbArray, int offset,
int scansize)
```

Sets an array of integer pixels in the default RGB color model (TYPE\_INT\_ARGB) and default sRGB color space, into a rectangular portion of the image data. There are only 8-bits of precision for each color component in the returned data when using this method. With a specified coordinate (x, y) in the this image, the ARGB pixel can be accessed in this way:

```
pixel = rgbArray[offset + (y-startY)*scansize + (x-startX)];
```

WARNING: No dithering takes place.

**Parameters:**

`startX` - the x-coordinate of the upper-left corner of the specified rectangular region

`startY` - the y-coordinate of the upper-left corner of the specified rectangular region

`w` - the width of the specified rectangular region

`h` - the height of the specified rectangular region

`rgbArray` - the ARGB pixels

`offset` - offset into the `rgbArray`

`scansize` - scanline stride for the `rgbArray`

**Since:**

MHP 1.0

---

**toString()**

```
public java.lang.String toString()
```

Returns a `String` representation of this `DVBBufferedImage` object and its values.

**Overrides:**

`java.lang.Object.toString()` in class `java.lang.Object`

**Returns:**

a `String` representing this `DVBBufferedImage`.

# org.dvb.ui DVBColor

## Syntax

```
public class DVBColor extends javax.tv.graphics.AlphaColor
```

```
java.lang.Object
|
+--java.awt.Color
|
+--javax.tv.graphics.AlphaColor
|
+--org.dvb.ui.DVBColor
```

## All Implemented Interfaces:

java.io.Serializable

## Description

A Color class which adds the notion of alpha. It is compatible with the JDK1.2 java.awt.Color class DVBColor extends org.davic.awt.Color which extends java.awt.Color. In implementations using the JDK1.1 this class adds support for alpha In implementations using the JDK1.2 the additional methods would just call super. Because DVBColor extends Color the signatures in the existing classes do not change. Classes like Component should work with DVBColor internally. Instances of this class are a container for the values which are passed in to the constructor. Any approximations made by the platform are made when the colors are used.

## Since:

MHP 1.0

## Constructors

---

### DVBColor(Color)

```
public DVBColor(java.awt.Color c)
```

Constructs a new DVBColor using the specified color. If c supports alpha, e.g. if it is an instance of javax.tv.graphics.AlphaColor or JDK 1.2's java.awt.Color, then the alpha value of c shall be used.

#### Parameters:

c - the java.awt.Color used to create a new DVBColor

---

### DVBColor(float, float, float, float)

```
public DVBColor(float r, float g, float b, float a)
```

Creates an sRGB color with the specified red, green, blue, and alpha values in the range (0.0 - 1.0). The actual color used in rendering will depend on finding the best match given the color space available for a given output device.

#### Parameters:

r - the red component

g - the green component

b - the blue component

a - the alpha component

**See Also:**

`java.awt.Color.getRed()`, `java.awt.Color.getGreen()`,  
`java.awt.Color.getBlue()`, `getAlpha()`, `getRGB()`

### DVBColor(int, boolean)

```
public DVBColor(int rgba, boolean hasalpha)
```

Creates an sRGB color with the specified combined RGBA value consisting of the alpha component in bits 24-31, the red component in bits 16-23, the green component in bits 8-15, and the blue component in bits 0-7. If the hasalpha argument is False, alpha is defaulted to 255.

**Parameters:**

rgba - the combined RGBA components

hasalpha - true if the alpha bits are valid, false otherwise

**See Also:**

`java.awt.Color.getRed()`, `java.awt.Color.getGreen()`,  
`java.awt.Color.getBlue()`, `getAlpha()`, `getRGB()`

### DVBColor(int, int, int, int)

```
public DVBColor(int r, int g, int b, int a)
```

Creates an sRGB color with the specified red, green, blue, and alpha values in the range (0 - 255).

**Parameters:**

r - the red component

g - the green component

b - the blue component

a - the alpha component

**See Also:**

`java.awt.Color.getRed()`, `java.awt.Color.getGreen()`,  
`java.awt.Color.getBlue()`, `getAlpha()`, `getRGB()`

## Methods

### brighter()

```
public java.awt.Color brighter()
```

Creates a brighter version of this color. This method applies an arbitrary scale factor to each of the three RGB components of the color to create a brighter version of the same color. Although brighter and darker are inverse operations, the results of a series of invocations of these two methods may be inconsistent because of rounding errors.

**Overrides:**

`javax.tv.graphics.AlphaColor.brighter()` in class `javax.tv.graphics.AlphaColor`

**Returns:**

a new DVBColor object (cast to a `java.awt.Color` object) representing a brighter version of this color. Applications can recast it to a `org.dvb.ui.DVBColor` object

**See Also:**

`java.awt.Color.brighter()`

---

**darker()**

```
public java.awt.Color darker()
```

Creates a darker version of this color. This method applies an arbitrary scale factor to each of the three RGB components of the color to create a darker version of the same color. Although `brighter` and `darker` are inverse operations, the results of a series of invocations of these two methods may be inconsistent because of rounding errors.

**Overrides:**

`javax.tv.graphics.AlphaColor.darker()` in class `javax.tv.graphics.AlphaColor`

**Returns:**

a new `DVBColor` object (cast to a `java.awt.Color` object), representing a darker version of this color. Applications can recast it to a `org.dvb.ui.DVBColor` object

**See Also:**

`java.awt.Color.darker()`

---

**equals(Object)**

```
public boolean equals(java.lang.Object obj)
```

Determines whether another object is equal to this color. The result is true if and only if the argument is not null and is a `DVBColor` object that has the same red, green, blue and alpha values as this object.

**Overrides:**

`javax.tv.graphics.AlphaColor.equals(java.lang.Object)` in class `javax.tv.graphics.AlphaColor`

**Parameters:**

`obj` -- the object to compare with.

**Returns:**

true if the objects are the same; false otherwise.

**Since:**

JDK1.0

---

**getAlpha()**

```
public int getAlpha()
```

Returns the alpha component. In the range 0-255.

**Overrides:**

`javax.tv.graphics.AlphaColor.getAlpha()` in class `javax.tv.graphics.AlphaColor`

**Returns:**

the alpha component

**See Also:**

`getRGB()`

---

**getRGB()**

```
public int getRGB()
```

Returns the RGB value representing the color in the default sRGB ColorModel. (Bits 24-31 are alpha, 16-23 are red, 8-15 are green, 0-7 are blue).

**Overrides:**

`javax.tv.graphics.AlphaColor.getRGB()` in class `javax.tv.graphics.AlphaColor`

**Returns:**

the RGB value representing the color in the default sRGB ColorModel.

**Since:**

JDK1.0

**See Also:**

`java.awt.Color.getRed()`, `java.awt.Color.getGreen()`,  
`java.awt.Color.getBlue()`, [getAlpha\(\)](#)

---

**toString()**

```
public java.lang.String toString()
```

Creates a string that represents this color and indicates the values of its ARGB components.

**Overrides:**

`javax.tv.graphics.AlphaColor.toString()` in class `javax.tv.graphics.AlphaColor`

**Returns:**

a representation of this color as a String object.

**Since:**

JDK1.0

# org.dvb.ui DVBGraphics

## Syntax

```
public abstract class DVBGraphics extends java.awt.Graphics
```

```
java.lang.Object
|
+--java.awt.Graphics
|
+--org.dvb.ui.DVBGraphics
```

## Description

The `DVBGraphics` class is a adapter class to support alpha compositing in an MHP device. Most methods directly delegate to `java.awt.Graphics` other methods could delegate to the appropriate methods in `java.awt.Graphics2D` where available or could be implemented in native code This class inherits from `java.awt.Graphics` in implementations using the JDK1.1. In implementations using the JDK1.2 `DVBGraphics` inherits from `java.awt.Graphics2D`. **In MHP devices all Graphics Objects are DVBGraphics objects.** Thus one can get a `DVBGraphics` by casting a given `Graphics` object. The normal compositing rule used is **DVBAlphaComposite.SRC**. This is the fastest rule because there is no computation. Note this is not the default behaviour of other graphics libraries. When drawing pictures with an alpha channel the transparent part will be transparent to the video in the background. Programmers should set the rule to `DVBAlphaComposite.SRC_OVER` when drawing images or other shapes which shall be transparent to the graphics in the background The intersection between `setDVBCompsite` in this class and the `setPaintMode` and `setXORMode` methods inherited from `java.awt.Graphics` shall be as follows.

- Calling `setPaintMode` on an instance of this class shall be equivalent to calling `setDVBComposite(DVBAlphaComposite.Src)`.
- Calling `setXORMode` on an instance of this class shall be equivalent to calling `setDVBComposite` with a special and implementation dependent `DVBAlphaComposite` object with implements the semantics specified for this method in the parent class.
- Calling `getDVBComposite` when `setXORMode` is the last `DVBComposite` set shall return this implementation dependent object. Conformant MHP applications shall not do anything with or to this object including calling any methods on it.
- This specification does not tighten, refine or detail the definition of the `setXORMode` beyond what is specified for the parent class.

Note: Implementations of XOR mode may change colours with alpha to without and vice versa (reversibly).

## Since:

MHP1.0

## See Also:

`java.awt.Graphics`

## Constructors

---

### DVBGraphics()

```
protected DVBGraphics()
```

Constructs a new `DVBGraphics` object. This constructor is the default constructor for a graphics context.

Since `DVBGraphics` is an abstract class, applications cannot call this constructor directly. `DVBGraphics` contexts are obtained from other `DVBGraphics` contexts or are created by casting `java.awt.Graphics` to `DVBGraphics`.

**Since:**

MHP 1.0

**See Also:**

`java.awt.Graphics.create()`, `java.awt.Component.getGraphics()`

## Methods

---

### **getAvailableCompositeRules()**

```
public abstract int[] getAvailableCompositeRules()
```

Returns all available Porter-Duff Rules for this specific Graphics context. E.g. a device could support the `SRC_OVER` rule when using a destination which does not have Alpha or where the alpha is null, while this rule is not available when drawing on a graphic context where the destination has alpha. Which rules are supported for the different graphics objects is defined in the Minimum Platform Capabilities of the MHP spec.

**Returns:**

all available Porter-Duff Rules for this specific Graphics context.

**Since:**

MHP 1.0

---

### **getBestColorMatch(Color)**

```
public DVBColor getBestColorMatch(java.awt.Color c)
```

Returns the best match for the specified `Color` as a `DVBColor`, in a device-dependent manner, as constrained by the MHP graphics reference model.

**Parameters:**

`c` - the specified `Color`.

**Returns:**

the best `DVBColor` match for the specified `Color`.

**Since:**

MHP 1.0

---

### **getColor()**

```
public abstract java.awt.Color getColor()
```

Gets this graphics context's current color. This will return a `DVBColor` cast to `java.awt.Color`.

**Overrides:**

`java.awt.Graphics.getColor()` in class `java.awt.Graphics`

**Returns:**

this graphics context's current color.

**Since:**



MHP 1.0

**See Also:**

[DVBColor](#), [java.awt.Color](#), [setColor\(Color\)](#)

**getDVBComposite()**

```
public abstract DVBAAlphaComposite getDVBComposite()
```

Returns the current `DVBAAlphaComposite` in the `DVBGraphics` context. This method could delegate to a `java.awt.Graphics2D` object where available

**Returns:**

the current `DVBGraphics` `DVBAAlphaComposite`, which defines a compositing style.

**Since:**

MHP 1.0

**See Also:**

[setDVBComposite\(DVBAAlphaComposite\)](#)

**getType()**

```
public int getType()
```

Returns the `Sample Model` (`DVBBufferedImage.TYPE_BASE`, `DVBBufferedImage.TYPE_ADVANCED`) which is used in the on/off screen buffer this graphics object draws into.

**Returns:**

the type of the `Sample Model`

**Since:**

MHP 1.0

**See Also:**

[DVBBufferedImage](#)

**setColor(Color)**

```
public abstract void setColor(java.awt.Color c)
```

Sets this graphics context's current color to the specified color. All subsequent graphics operations using this graphics context use this specified color. Note that color `c` can be a `DVBColor`

**Overrides:**

`java.awt.Graphics.setColor(java.awt.Color)` in class `java.awt.Graphics`

**Parameters:**

`c` - the new rendering color.

**Since:**

MHP 1.0

**See Also:**

[java.awt.Color](#), [DVBColor](#), [getColor\(\)](#)

**setDVBComposite(DVBAAlphaComposite)**

```
public abstract void setDVBComposite(DVBAAlphaComposite comp)
```

Sets the `DVBAAlphaComposite` for the `DVBGraphics` context. The `DVBAAlphaComposite` is used in all drawing methods such as `drawImage`, `drawString`, `draw`, and `fill`. It specifies how new

pixels are to be combined with the existing pixels on the graphics device during the rendering process.

This method could delegate to a Graphics2D object or to an native implementation

**Parameters:**

`comp` - the `DVBAlphaComposite` object to be used for rendering

**Throws:**

`UnsupportedDrawingOperationException` - when the requested Porter-Duff rule is not supported by this graphics context

**Since:**

MHP 1.0

**See Also:**

`java.awt.Graphics.setXORMode(Color)`, `java.awt.Graphics.setPaintMode()`, `DVBAlphaComposite`

---

**toString()**

```
public java.lang.String toString()
```

Returns a `String` object representing this `DVBGraphics` object's value.

**Overrides:**

`java.awt.Graphics.toString()` in class `java.awt.Graphics`

**Returns:**

a string representation of this graphics context.

**Since:**

MHP 1.0

# org.dvb.ui DVBRasterFormatException

## Syntax

```
public class DVBRasterFormatException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
|
+--java.lang.Exception
|
+--org.dvb.ui.DVBRasterFormatException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This exception is thrown for some invalid operations on instances of `DVBBufferedImage`. The precise conditions are defined in the places where this exception is throw.

## Since:

MHP 1.0.1

## See Also:

[DVBBufferedImage](#)

## Constructors

---

### DVBRasterFormatException(String)

```
public DVBRasterFormatException(java.lang.String s)
```

Constructs an instance of `DVBRasterFormatException` with the specified detail message.

#### Parameters:

`s` - the detail message

#### Since:

MHP1.0

# org.dvb.ui DVBTextLayoutManager

## Syntax

public class DVBTextLayoutManager implements [HTextLayoutManager](#)

```
java.lang.Object
|
+--org.dvb.ui.DVBTextLayoutManager
```

## All Implemented Interfaces:

[HTextLayoutManager](#)

## Description

The DVBTextLayoutManager provides a text rendering layout mechanism for the org.havi.ui.HStaticText org.havi.ui.HText and org.havi.ui.HTextButton classes.

The semantics of the rendering behaviour and the settings are specified in the "Text presentation" annex of this specification. The DVBTextLayoutManager renders the text according to the semantics described in that annex.

## Fields

---

### HORIZONTAL\_CENTER

```
public static final int HORIZONTAL_CENTER
```

The text should be centered horizontally.

---

### HORIZONTAL\_END\_ALIGN

```
public static final int HORIZONTAL_END_ALIGN
```

The text should be horizontally to the horizontal end side (e.g. when start corner is upper left and line orientation horizontal, meaning text that is read left to right from top to bottom, this implies alignment to right).

---

### HORIZONTAL\_START\_ALIGN

```
public static final int HORIZONTAL_START_ALIGN
```

The text should be aligned horizontally to the horizontal start side (e.g. when start corner is upper left and line orientation horizontal, meaning text that is read left to right from top to bottom, this implies alignment to left).

---

### LINE\_ORIENTATION\_HORIZONTAL

```
public static final int LINE_ORIENTATION_HORIZONTAL
```

Horizontal line orientation.

---

**LINE\_ORIENTATION\_VERTICAL**

```
public static final int LINE_ORIENTATION_VERTICAL
```

Vertical line orientation.

---

**START\_CORNER\_LOWER\_LEFT**

```
public static final int START_CORNER_LOWER_LEFT
```

Lower left text start corner.

---

**START\_CORNER\_LOWER\_RIGHT**

```
public static final int START_CORNER_LOWER_RIGHT
```

Lower right text start corner.

---

**START\_CORNER\_UPPER\_LEFT**

```
public static final int START_CORNER_UPPER_LEFT
```

Upper left text start corner.

---

**START\_CORNER\_UPPER\_RIGHT**

```
public static final int START_CORNER_UPPER_RIGHT
```

Upper right text start corner.

---

**VERTICAL\_CENTER**

```
public static final int VERTICAL_CENTER
```

The text should be centered vertically.

---

**VERTICAL\_END\_ALIGN**

```
public static final int VERTICAL_END_ALIGN
```

The text should be aligned vertically to the vertical end side (e.g. when start corner is upper left and line orientation horizontal, meaning text that is read left to right from top to bottom, this implies alignment to bottom).

This is defined by the section "Vertical limits" in the "Text presentation" annex of this specification.

---

**VERTICAL\_START\_ALIGN**

```
public static final int VERTICAL_START_ALIGN
```

The text should be aligned vertically to the vertical start side (e.g. when start corner is upper left and line orientation horizontal, meaning text that is read left to right from top to bottom, this implies alignment to top).

This is defined by the section "Vertical limits" in the "Text presentation" annex of this specification.

## Constructors

---

**DVBTextLayoutManager()**

```
public DVBTextLayoutManager()
```

Constructs a DVBTextLayoutManager object with default parameters (HORIZONTAL\_START\_ALIGN, VERTICAL\_START\_ALIGN, LINE\_ORIENTATION\_HORIZONTAL, START\_CORNER\_UPPER\_LEFT, wrap = true, linespace = (point size of the default font for HVisible) + 7, letterspace = 0, horizontalTabSpace = 56)

---

**DVBTextLayoutManager(int, int, int, int, boolean, int, int, int)**

```
public DVBTextLayoutManager(int horizontalAlign, int verticalAlign, int lineOrientation,
    int startCorner, boolean wrap, int linespace, int letterspace,
    int horizontalTabSpace)
```

Constructs a DVBTextLayoutManager object.

**Parameters:**

horizontalAlign - Horizontal alignment setting

verticalAlign - Vertical alignment setting

lineOrientation - Line orientation setting

startCorner - Starting corner setting

wrap - Text wrapping setting

linespace - Line spacing setting expressed in points

letterspace - Letterspacing adjustment relative to the default letterspacing. Expressed in units of 1/256th point as the required increase in the spacing between consecutive characters. May be either positive or negative.

horizontalTabSpace - Horizontal tabulation setting in points

## Methods

---

**addTextOverflowListener(TextOverflowListener)**

```
public void addTextOverflowListener(TextOverflowListener l)
```

Register a TextOverflowListener that will be notified if the text string does not fit in the component when rendering.

**Parameters:**

l - a listener object

---

**getHorizontalAlign()**

```
public int getHorizontalAlign()
```

Get the horizontal alignment.

**Returns:**

Horizontal alignment setting

---

**getHorizontalTabSpacing()**

```
public int getHorizontalTabSpacing()
```

Get the horizontal tabulation spacing.

**Returns:**

the horizontal tabulation spacing

---

**getInsets()**

```
public java.awt.Insets getInsets()
```

Returns the insets that this text layout manager uses. When rendering text, it leaves empty margins of the size defined by the insets. The actual area used for the text is the area of the component decreased by the amount of insets at each edge. The default insets, if not set explicitly using `setInsets`, are 0 at each edge, i.e. no margins.

**Returns:**

Insets used by this text layout manager.

---

**getLetterSpace()**

```
public int getLetterSpace()
```

Get the letter space setting.

**Returns:**

letter space setting

---

**getLineOrientation()**

```
public int getLineOrientation()
```

Get the line orientation.

**Returns:**

Line orientation setting

---

**getLineSpace()**

```
public int getLineSpace()
```

Get the line space setting.

**Returns:**

line space setting or -1, if the default line spacing is determined from the size of the default font is used.

---

**getStartCorner()**

```
public int getStartCorner()
```

Get the starting corner.

**Returns:**

Starting corner setting

---

**getTextWrapping()**

```
public boolean getTextWrapping()
```

Get the text wrapping setting.

**Returns:**

text wrapping setting

---

**getVerticalAlign()**

```
public int getVerticalAlign()
```

Get the vertical alignment.

**Returns:**

Vertical alignment setting

---

**removeTextOverflowListener(TextOverflowListener)**

```
public void removeTextOverflowListener(TextOverflowListener l)
```

Removes a TextOverflowListener that has been registered previously.

**Parameters:**

`l` - a listener object

---

**render(String, Graphics, HVisible, Insets)**

```
public void render(java.lang.String markedUpString, java.awt.Graphics g, HVisible v,  
                  java.awt.Insets insets)
```

Render the string. The `HTextLayoutManager` should use the passed `HVisible` object to determine any additional information required to render the string, e.g. `Font`, `Color` etc.

The text should be laid out in the layout area, which is defined by the bounds of the specified `HVisible`, after subtracting the insets. If the insets are `null` the full bounding rectangle is used as the area to render text into.

The `HTextLayoutManager` should not modify the clipping rectangle of the `Graphics` object.

**Specified By:**

`render(String, Graphics, HVisible, Insets)` in interface `HTextLayoutManager`

**Parameters:**

`markedUpString` - the string to render.

`g` - the graphics context, including a clipping rectangle which encapsulates the area within which rendering is permitted. If a valid insets value is passed to this method then text must only be rendered into the bounds of the widget after the insets are subtracted. If the insets value is `null` then text is rendered into the entire bounding area of the `HVisible`. It is implementation specific whether or not the renderer takes into account the intersection of the clipping rectangle in each case for optimization purposes.

`v` - the `HVisible` into which to render.

`insets` - the insets to determine the area in which to layout the text, or `null`.

---

**setHorizontalAlign(int)**

```
public void setHorizontalAlign(int horizontalAlign)
```

Set the horizontal alignment.

**Parameters:**

`horizontalAlign` - Horizontal alignment setting



---

**setHorizontalTabSpacing(int)**

```
public void setHorizontalTabSpacing(int horizontalTabSpace)
```

Set the horizontal tabulation spacing.

**Parameters:**

horizontalTabSpace - tab spacing in points

---

**setInsets(Insets)**

```
public void setInsets(java.awt.Insets insets)
```

Sets the insets that should be used by this text layout manager. The text is rendered to the area defined by the area of the component decreased by the amount of insets at each edge. If this method is not called, the default insets are 0 at each edge.

**Parameters:**

insets - Insets that should be used

---

**setLetterSpace(int)**

```
public void setLetterSpace(int letterSpace)
```

Set the letter space setting.

**Parameters:**

letterSpace - letter space setting

---

**setLineOrientation(int)**

```
public void setLineOrientation(int lineOrientation)
```

Set the line orientation.

**Parameters:**

lineOrientation - Line orientation setting

---

**setLineSpace(int)**

```
public void setLineSpace(int lineSpace)
```

Set the line space setting. Using -1 as the line space setting shall cause the line spacing to be determined from the size of the default font.

**Parameters:**

lineSpace - line space setting

---

**setStartCorner(int)**

```
public void setStartCorner(int startCorner)
```

Set the starting corner.

**Parameters:**

startCorner - Starting corner setting

---

**setTextWrapping(boolean)**

```
public void setTextWrapping(boolean wrap)
```

Set the text wrapping setting

**Parameters:**

`wrap` - Text wrapping setting

---

**setVerticalAlign(int)**

```
public void setVerticalAlign(int verticalAlign)
```

Set the vertical alignment.

**Parameters:**

`verticalAlign` - Vertical alignment setting

# org.dvb.ui FontFactory

## Syntax

```
public class FontFactory
    java.lang.Object
    |
    +--org.dvb.ui.FontFactory
```

## Description

Provides a mechanism for applications to instantiate fonts that are not built into the system.

## Constructors

---

### FontFactory()

```
public FontFactory()
```

Constructs a `FontFactory` for the font index file bound to this application in the application signaling. The call to the constructor is synchronous and shall block until the font index file has been retrieved or an an exception is thrown.

#### Throws:

`FontFormatException` - if there is an error in the font index file bound with the application.

`java.io.IOException` - if there is no font index file bound with the application, or if there is an error attempting to access the data in that file.

---

### FontFactory(URL)

```
public FontFactory(java.net.URL u)
```

Constructs a `FontFactory` for the specification of fonts found at the given location. The call to the constructor is synchronous and shall block until the font file has been retrieved or an exception is thrown.

#### Parameters:

`u` - The source of the font description

#### Throws:

`java.io.IOException` - if there is an error attempting to access the data referenced by the URL

`IllegalArgumentException` - if the URL is not both valid and supported

`SecurityException` - if access to the specified URL is denied by security policy

`FontFormatException` - if the file at that URL is not a valid font file as specified in the main body of this specification

## Methods

---

**createFont(String, int, int)**

```
public java.awt.Font createFont(java.lang.String name, int style, int size)
```

Creates a font object from the font source associated with this `FontFactory`. This font will remain valid even if the `FontFactory` is no longer reachable from application code. The name returned by `Font.getName()` might not be the same as the name supplied, for example, it might have a string prepended to it that identifies the source `FontFactory` in a platform-dependant manner. For `FontFactory` instances bound to the font index file of an application, the call to the method is synchronous and shall block until either an exception is thrown or any required network access has completed.

The value of the style argument must be as defined in `java.awt.Font`. Valid values are the following:

- `java.awt.Font.PLAIN`
- `java.awt.Font.BOLD`
- `java.awt.Font.ITALIC`
- `java.awt.Font.BOLD + java.awt.Font.ITALIC`

**Parameters:**

`name` - the font name

`style` - the constant style used, such as `java.awt.Font.PLAIN`.

`size` - the point size of the font

**Throws:**

`FontNotAvailableException` - if a font with given parameters cannot be located or created.

`java.io.IOException` - if there is an error retrieving a font from the network. Thrown only for font factory instances bound to the font index file of an application.

`IllegalArgumentException` - if the style parameter is not in the set of valid values, or if the size parameter is zero or negative.

`FontFormatException` - if the font file is not a valid font file as specified in the main body of this specification. Thrown only for font factory instances bound to the font index file of an application.

# org.dvb.ui FontFormatException

## Syntax

```
public class FontFormatException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
   |
   +--java.lang.Exception
      |
      +--org.dvb.ui.FontFormatException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Thrown when attempt is made to read a file describing a font when the contents of that file are not valid.

## Constructors

---

### FontFormatException()

```
public FontFormatException()
```

Constructs a `FontNotAvailableException` with `null` as its error detail message.

---

### FontFormatException(String)

```
public FontFormatException(java.lang.String s)
```

Constructs a `FontFormatException` with the specified detail message. The error message string `s` can later be retrieved by the `java.lang.Throwable.getMessage()` method of class `java.lang.Throwable`.

#### Parameters:

`s` - the detail message.

# org.dvb.ui FontNotAvailableException

## Syntax

```
public class FontNotAvailableException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
   |
   +--java.lang.Exception
      |
      +--org.dvb.ui.FontNotAvailableException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Thrown when attempt is made to instantiate a font that cannot be located.

## Constructors

---

### FontNotAvailableException()

```
public FontNotAvailableException()
```

Constructs a `FontNotAvailableException` with `null` as its error detail message.

---

### FontNotAvailableException(String)

```
public FontNotAvailableException(java.lang.String s)
```

Constructs a `FontNotAvailableException` with the specified detail message. The error message string `s` can later be retrieved by the `java.lang.Throwable.getMessage()` method of class `java.lang.Throwable`.

#### Parameters:

`s` - the detail message.

# org.dvb.ui TestOpacity

## Syntax

```
public interface TestOpacity
```

## All Known Implementing Classes:

```
HContainer, HComponent
```

## Description

Interface implemented by Components or Containers in order to allow the platform to query whether their paint method is fully opaque.

## Methods

---

### isOpaque()

```
public boolean isOpaque()
```

Returns true if the entire area of the component as given by the `getBounds` method, is fully opaque. Hence its paint method (or surrogate methods) guarantees that all pixels are painted in an opaque Color.

Classes implementing this interface shall return true from their implementation of this method if and only if their implementation can guarantee full opacity. The consequences of an invalid overridden value are implementation specific.

#### Returns:

true if all the pixels with the `java.awt.Component#getBounds` method are fully opaque, otherwise false.

org.dvb.ui

# TextOverflowListener

## Syntax

```
public interface TextOverflowListener
```

## Description

The `TextOverflowListener` is an interface that an application may implement and register in the `DVBTextLayoutManager`. This listener will be notified if the text string does not fit within the component when rendering it.

## Methods

---

### `notifyTextOverflow(String, HVisible, boolean, boolean)`

```
public void notifyTextOverflow(java.lang.String markedUpString, HVisible v,  
                               boolean overflowedHorizontally, boolean overflowedVertically)
```

This method is called by the `DVBTextLayoutManager` if the text does not fit within the component

#### Parameters:

`markedUpString` - the string that was rendered

`v` - the `HVisible` object that was being rendered

`overflowedHorizontally` - true if the text overflow the bounds of the component in the horizontal direction; otherwise false

`overflowedVertically` - true if the text overflow the bounds of the component in the vertical direction; otherwise false



org.dvb.ui

# UnsupportedDrawingOperationException

## Syntax

```
public class UnsupportedDrawingOperationException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--org.dvb.ui.UnsupportedDrawingOperationException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

The `UnsupportedDrawingOperationException` class represents an exception that is thrown if an drawing operation is not supported on this platform. E.g. `DVBGraphics.setComposite` could throw an Exception when setting the `SRC_OVER` rule on some devices while the `SRC` rule will always work.

## Since:

MHP 1.0

## Constructors

---

### UnsupportedDrawingOperationException(String)

```
public UnsupportedDrawingOperationException(java.lang.String s)
```

Constructs an instance of `UnsupportedDrawingOperationException` with the specified detail message.

#### Parameters:

`s` - the detail message

#### Since:

MHP1.0

# Annex V (normative): HAVi Level 2 User Interface

## V.1 Register of Constants

```

public final static int org.havi.ui.HAdjustableLook.ADJUST_NONE = 0;
public final static int org.havi.ui.HAdjustableLook.ADJUST_BUTTON_LESS = 1;
public final static int org.havi.ui.HAdjustableLook.ADJUST_BUTTON_MORE = 2;
public final static int org.havi.ui.HAdjustableLook.ADJUST_PAGE_LESS = 2;
public final static int org.havi.ui.HAdjustableLook.ADJUST_PAGE_MORE = 3;
public final static int org.havi.ui.HAdjustableLook.ADJUST_THUMB = 4;

public final static int org.havi.ui.HAnimateEffect.PLAY_REPEATING = 1;
public final static int org.havi.ui.HAnimateEffect.PLAY_ALTERNATING = 2;
public final static int org.havi.ui.HAnimateEffect.REPEAT_INFINITE = -1;

public final static int org.havi.ui.HbackgroundConfigTemplate.CHANGEABLE_SINGLE_COLOR = 0x0A;
public final static int org.havi.ui.HbackgroundConfigTemplate.STILL_IMAGE = 0x0B;

public final static int org.havi.ui.HFontCapabilities.BASIC_LATIN = 1;
public final static int org.havi.ui.HFontCapabilities.LATIN_1_SUPPLEMENT = 2;
public final static int org.havi.ui.HFontCapabilities.LATIN_EXTENDED_A = 3;
public final static int org.havi.ui.HFontCapabilities.LATIN_EXTENDED_B = 4;
public final static int org.havi.ui.HFontCapabilities.IPA_EXTENSIONS = 5;
public final static int org.havi.ui.HFontCapabilities.SPACING_MODIFIER_LETTERS = 6;
public final static int org.havi.ui.HFontCapabilities.COMBINING_DIACRITICAL_MARKS = 7;
public final static int org.havi.ui.HFontCapabilities.BASIC_GREEK = 8;
public final static int org.havi.ui.HFontCapabilities.GREEK_SYMBOLS_AND_COPTIC = 9;
public final static int org.havi.ui.HFontCapabilities.CYRILLIC = 10;
public final static int org.havi.ui.HFontCapabilities.ARMEIAN = 11;
public final static int org.havi.ui.HFontCapabilities.BASIC_HEBREW = 12;
public final static int org.havi.ui.HFontCapabilities.HEBREW_EXTENDED = 13;
public final static int org.havi.ui.HFontCapabilities.BASIC_ARABIC = 14;
public final static int org.havi.ui.HFontCapabilities.ARABIC_EXTENDED = 15;
public final static int org.havi.ui.HFontCapabilities.DEVANAGARI = 16;
public final static int org.havi.ui.HFontCapabilities.BENGALI = 17;
public final static int org.havi.ui.HFontCapabilities.GURMUKHI = 18;
public final static int org.havi.ui.HFontCapabilities.GUJARATI = 19;
public final static int org.havi.ui.HFontCapabilities.ORIYA = 20;
public final static int org.havi.ui.HFontCapabilities.TAMIL = 21;
public final static int org.havi.ui.HFontCapabilities.TELUGU = 22;
public final static int org.havi.ui.HFontCapabilities.KANNADA = 23;
public final static int org.havi.ui.HFontCapabilities.MALAYALAM = 24;
public final static int org.havi.ui.HFontCapabilities.THAI = 25;
public final static int org.havi.ui.HFontCapabilities.LAO = 26;
public final static int org.havi.ui.HFontCapabilities.BASIC_GEORGIAN = 27;
public final static int org.havi.ui.HFontCapabilities.GEORGIAN_EXTENDED = 28;
public final static int org.havi.ui.HFontCapabilities.HANGUL_JAMO = 29;
public final static int org.havi.ui.HFontCapabilities.LATIN_EXTENDED_ADDITIONAL = 30;
public final static int org.havi.ui.HFontCapabilities.GREEK_EXTENDED = 31;
public final static int org.havi.ui.HFontCapabilities.GENERAL_PUNCTUATION = 32;
public final static int org.havi.ui.HFontCapabilities.SUPERSCRIPTS_AND_SUBSCRIPTS = 33;
public final static int org.havi.ui.HFontCapabilities.CURRENCY_SYMBOLS = 34;
public final static int org.havi.ui.HFontCapabilities.COMBINING_DIACRITICAL_MARKS_FOR_SYMBOLS = 35;
public final static int org.havi.ui.HFontCapabilities.LETTERLIKE_SYMBOLS = 36;
public final static int org.havi.ui.HFontCapabilities.NUMBER_FORMS = 37;
public final static int org.havi.ui.HFontCapabilities.ARROWS = 38;
public final static int org.havi.ui.HFontCapabilities.MATHEMATICAL_OPERATORS = 39;
public final static int org.havi.ui.HFontCapabilities.MISCELLANEOUS_TECHNICAL = 40;
public final static int org.havi.ui.HFontCapabilities.CONTROL_PICTURES = 41;
public final static int org.havi.ui.HFontCapabilities.OPTICAL_CHARACTER_RECOGNITION = 42;
public final static int org.havi.ui.HFontCapabilities.ENCLOSED_ALPHANUMERICS = 43;
public final static int org.havi.ui.HFontCapabilities.BOX_DRAWING = 44;
public final static int org.havi.ui.HFontCapabilities.BLOCK_ELEMENTS = 45;
public final static int org.havi.ui.HFontCapabilities.GEOMETRICAL_SHAPES = 46;
public final static int org.havi.ui.HFontCapabilities.MISCELLANEOUS_SYMBOLS = 47;
public final static int org.havi.ui.HFontCapabilities.DINGBATS = 48;
public final static int org.havi.ui.HFontCapabilities.CJK_SYMBOLS_AND_PUNCTUATION = 49;
public final static int org.havi.ui.HFontCapabilities.HIRAGANA = 50;
public final static int org.havi.ui.HFontCapabilities.KATAKANA = 51;

```

```

public final static int org.havi.ui.HFontCapabilities.BOPOMOFO = 52;
public final static int org.havi.ui.HFontCapabilities.HANGUL_COMPATIBILITY_JAMO = 53;
public final static int org.havi.ui.HFontCapabilities.CJK_MISCELLANEOUS = 54;
public final static int org.havi.ui.HFontCapabilities.ENCLOSED_CJK_LETTERS_AND_MONTHS = 55;
public final static int org.havi.ui.HFontCapabilities.CJK_COMPATIBILITY = 56;
public final static int org.havi.ui.HFontCapabilities.HANGUL = 57;
public final static int org.havi.ui.HFontCapabilities.HANGUL_SUPPLEMENTARY_A = 58;
public final static int org.havi.ui.HFontCapabilities.HANGUL_SUPPLEMENTARY_B = 59;
public final static int org.havi.ui.HFontCapabilities.CJK_UNIFIED_IDEOGRAPHS = 60;
public final static int org.havi.ui.HFontCapabilities.PRIVATE_USE_AREA = 61;
public final static int org.havi.ui.HFontCapabilities.CJK_COMPATIBILITY_IDEOGRAPHS = 62;
public final static int org.havi.ui.HFontCapabilities.ALPHABETIC_PRESENTATION_FORMS_A = 63;
public final static int org.havi.ui.HFontCapabilities.ARABIC_PRESENTATION_FORMS_A = 64;
public final static int org.havi.ui.HFontCapabilities.COMBINING_HALF_MARKS = 65;
public final static int org.havi.ui.HFontCapabilities.CJK_COMPATIBILITY_FORMS = 66;
public final static int org.havi.ui.HFontCapabilities.SMALL_FORM_VARIANTS = 67;
public final static int org.havi.ui.HFontCapabilities.ARABIC_PRESENTATION_FORMS_B = 68;
public final static int org.havi.ui.HFontCapabilities.HALFWIDTH_AND_FULLWIDTH_FORMS = 69;
public final static int org.havi.ui.HFontCapabilities.SPECIALS = 70;

public final static int org.havi.ui.HGraphicsConfigTemplate.VIDEO_MIXING = 0x0C;
public final static int org.havi.ui.HGraphicsConfigTemplate.MATTE_SUPPORT = 0x0D;
public final static int org.havi.ui.HGraphicsConfigTemplate.IMAGE_SCALING_SUPPORT = 0x0E;

public final static int org.havi.ui.HImageHints.NATURAL_IMAGE = 0x01;
public final static int org.havi.ui.HImageHints.CARTOON = 0x02;
public final static int org.havi.ui.HImageHints.BUSINESS_GRAPHICS = 0x03;
public final static int org.havi.ui.HImageHints.LINE_ART = 0x04;

public final static int org.havi.ui.HKeyboardInputPreferred.INPUT_ALPHANUMERIC = 0;
public final static int org.havi.ui.HKeyboardInputPreferred.INPUT_NUMERIC = 1;
public final static int org.havi.ui.HKeyboardInputPreferred.INPUT_ALPHA = 2;
public final static int org.havi.ui.HKeyboardInputPreferred.INPUT_CUSTOMIZED = 3;
public final static int org.havi.ui.HKeyboardInputPreferred.INPUT_ANY = 4;

public final static int org.havi.ui.HListGroup.ITEM_NOT_FOUND = -1;
public final static int org.havi.ui.HListGroup.ADD_INDEX_END = -1;
public final static int org.havi.ui.HListGroup.DEFAULT_LABEL_WIDTH = -1;
public final static int org.havi.ui.HListGroup.DEFAULT_LABEL_HEIGHT = -2;
public final static int org.havi.ui.HListGroup.DEFAULT_ICON_WIDTH = -3;
public final static int org.havi.ui.HListGroup.DEFAULT_ICON_HEIGHT = -4;

public final static int org.havi.ui.HOrientable.ORIENT_LEFT_TO_RIGHT = 0;
public final static int org.havi.ui.HOrientable.ORIENT_RIGHT_TO_LEFT = 1;
public final static int org.havi.ui.HOrientable.ORIENT_TOP_TO_BOTTOM = 2;
public final static int org.havi.ui.HOrientable.ORIENT_BOTTOM_TO_TOP = 3;

public final static int org.havi.ui.HScene.IMAGE_NONE = 0;
public final static int org.havi.ui.HScene.IMAGE_STRETCH = 1;
public final static int org.havi.ui.HScene.IMAGE_CENTER = 2;
public final static int org.havi.ui.HScene.IMAGE_TILE = 3;
public final static int org.havi.ui.HScene.NO_BACKGROUND_FILL = 0;
public final static int org.havi.ui.HScene.BACKGROUND_FILL = 1;

public final static int org.havi.ui.HSceneTemplate.REQUIRED = 0x01;
public final static int org.havi.ui.HSceneTemplate.PREFERRED = 0x02;
public final static int org.havi.ui.HSceneTemplate.UNNECESSARY = 0x03;
public final static int org.havi.ui.HSceneTemplate.GRAPHICS_CONFIGURATION = 0x00;
public final static int org.havi.ui.HSceneTemplate.SCENE_PIXEL_DIMENSION = 0x01;
public final static int org.havi.ui.HSceneTemplate.SCENE_PIXEL_LOCATION = 0x02;
public final static int org.havi.ui.HSceneTemplate.SCENE_SCREEN_DIMENSION = 0x04;
public final static int org.havi.ui.HSceneTemplate.SCENE_SCREEN_LOCATION = 0x08;

public final static int org.havi.ui.HScreenConfigTemplate.REQUIRED = 0x01;
public final static int org.havi.ui.HScreenConfigTemplate.PREFERRED = 0x02;
public final static int org.havi.ui.HScreenConfigTemplate.DONT_CARE = 0x03;
public final static int org.havi.ui.HScreenConfigTemplate.PREFERRED_NOT = 0x04;
public final static int org.havi.ui.HScreenConfigTemplate.REQUIRED_NOT = 0x05;
public final static int org.havi.ui.HScreenConfigTemplate.ZERO_BACKGROUND_IMPACT = 0x01;
public final static int org.havi.ui.HScreenConfigTemplate.ZERO_GRAPHICS_IMPACT = 0x02;
public final static int org.havi.ui.HScreenConfigTemplate.ZERO_VIDEO_IMPACT = 0x03;

```

```

public final static int org.havi.ui.HScreenConfigTemplate.INTERLACED_DISPLAY = 0x04;
public final static int org.havi.ui.HScreenConfigTemplate.FLICKER_FILTERING = 0x05;
public final static int org.havi.ui.HScreenConfigTemplate.VIDEO_GRAPHICS_PIXEL_ALIGNED = 0x06;
public final static int org.havi.ui.HScreenConfigTemplate.PIXEL_ASPECT_RATIO = 0x07;
public final static int org.havi.ui.HScreenConfigTemplate.PIXEL_RESOLUTION = 0x08;
public final static int org.havi.ui.HScreenConfigTemplate.SCREEN_RECTANGLE = 0x09;

public final static int org.havi.ui.HState.FOCUSED_STATE_BIT = 0x01;
public final static int org.havi.ui.HState.ACTIONED_STATE_BIT = 0x02;
public final static int org.havi.ui.HState.DISABLED_STATE_BIT = 0x04;
public final static int org.havi.ui.HState.FIRST_STATE = 0x80;
public final static int org.havi.ui.HState.NORMAL_STATE = 0x80;
public final static int org.havi.ui.HState.FOCUSED_STATE = 0x81;
public final static int org.havi.ui.HState.ACTIONED_STATE = 0x82;
public final static int org.havi.ui.HState.ACTIONED_FOCUSED_STATE = 0x83;
public final static int org.havi.ui.HState.DISABLED_STATE = 0x84;
public final static int org.havi.ui.HState.DISABLED_FOCUSED_STATE = 0x85;
public final static int org.havi.ui.HState.DISABLED_ACTIONED_STATE = 0x86;
public final static int org.havi.ui.HState.DISABLED_ACTIONED_FOCUSED_STATE = 0x87;
public final static int org.havi.ui.HState.ALL_STATES = 0x07;
public final static int org.havi.ui.HState.LAST_STATE = 0x07;

public final static int org.havi.ui.HStaticRange.SLIDER_BEHAVIOR = 0;
public final static int org.havi.ui.HStaticRange.SCROLLBAR_BEHAVIOR = 1;

public final static int org.havi.ui.HVideoConfigTemplate.GRAPHICS_MIXING = 0x0F;

public final static int org.havi.ui.HVisible.HALIGN_LEFT = 0;
public final static int org.havi.ui.HVisible.HALIGN_CENTER = 1;
public final static int org.havi.ui.HVisible.HALIGN_RIGHT = 2;
public final static int org.havi.ui.HVisible.HALIGN_JUSTIFY = 3;
public final static int org.havi.ui.HVisible.VALIGN_TOP = 0;
public final static int org.havi.ui.HVisible.VALIGN_CENTER = 4;
public final static int org.havi.ui.HVisible.VALIGN_BOTTOM = 8;
public final static int org.havi.ui.HVisible.VALIGN_JUSTIFY = 12;
public final static int org.havi.ui.HVisible.RESIZE_NONE = 0;
public final static int org.havi.ui.HVisible.RESIZE_PRESERVE_ASPECT = 1;
public final static int org.havi.ui.HVisible.RESIZE_ARBITRARY = 2;
public final static int org.havi.ui.HVisible.NO_BACKGROUND_FILL = 0;
public final static int org.havi.ui.HVisible.BACKGROUND_FILL = 1;
public final static int org.havi.ui.HVisible.FIRST_CHANGE = 0;
public final static int org.havi.ui.HVisible.TEXT_CONTENT_CHANGE = 0;
public final static int org.havi.ui.HVisible.GRAPHIC_CONTENT_CHANGE = 1;
public final static int org.havi.ui.HVisible.ANIMATE_CONTENT_CHANGE = 2;
public final static int org.havi.ui.HVisible.CONTENT_CHANGE = 3;
public final static int org.havi.ui.HVisible.STATE_CHANGE = 4;
public final static int org.havi.ui.HVisible.CARET_POSITION_CHANGE = 5;
public final static int org.havi.ui.HVisible.ECHO_CHAR_CHANGE = 6;
public final static int org.havi.ui.HVisible.EDIT_MODE_CHANGE = 7;
public final static int org.havi.ui.HVisible.MIN_MAX_CHANGE = 8;
public final static int org.havi.ui.HVisible.THUMB_OFFSETS_CHANGE = 9;
public final static int org.havi.ui.HVisible.ADJUSTMENT_VALUE_CHANGE = 13;
public final static int org.havi.ui.HVisible.ORIENTATION_CHANGE = 10;
public final static int org.havi.ui.HVisible.TEXT_VALUE_CHANGE = 11;
public final static int org.havi.ui.HVisible.ITEM_VALUE_CHANGE = 12;
public final static int org.havi.ui.HVisible.LIST_CONTENT_CHANGE = 14;
public final static int org.havi.ui.HVisible.LIST_ICONSIZE_CHANGE = 15;
public final static int org.havi.ui.HVisible.LIST_LABELSIZE_CHANGE = 16;
public final static int org.havi.ui.HVisible.LIST_MULTISELECTION_CHANGE = 17;
public final static int org.havi.ui.HVisible.LIST_SCROLLPOSITION_CHANGE = 18;
public final static int org.havi.ui.HVisible.UNKNOWN_CHANGE = 19;
public final static int org.havi.ui.HVisible.LAST_CHANGE = 19;
public final static int org.havi.ui.HVisible.NO_DEFAULT_WIDTH = -1;
public final static int org.havi.ui.HVisible.NO_DEFAULT_HEIGHT = -1;

public final static int org.havi.ui.event.HAdjustmentEvent.ADJUST_FIRST = 2000;
public final static int org.havi.ui.event.HAdjustmentEvent.ADJUST_LAST = 2005;
public final static int org.havi.ui.event.HAdjustmentEvent.ADJUST_START_CHANGE = 2000;
public final static int org.havi.ui.event.HAdjustmentEvent.ADJUST_LESS = 2001;
public final static int org.havi.ui.event.HAdjustmentEvent.ADJUST_MORE = 2002;
public final static int org.havi.ui.event.HAdjustmentEvent.ADJUST_PAGE_LESS = 2003;

```

```

public final static int org.havi.ui.event.HAdjustmentEvent.ADJUST_PAGE_MORE = 2004;
public final static int org.havi.ui.event.HAdjustmentEvent.ADJUST_END_CHANGE = 2005;

public final static int org.havi.ui.event.HBackgroundImageEvent.BACKGROUNDIMAGE_FIRST = 1;
public final static int org.havi.ui.event.HBackgroundImageEvent.BACKGROUNDIMAGE_LOADED = 1;
public final static int org.havi.ui.event.HBackgroundImageEvent.BACKGROUNDIMAGE_FILE_NOT_FOUND = 2;
public final static int org.havi.ui.event.HBackgroundImageEvent.BACKGROUNDIMAGE_IOERROR = 3;
public final static int org.havi.ui.event.HBackgroundImageEvent.BACKGROUNDIMAGE_INVALID = 4;
public final static int org.havi.ui.event.HBackgroundImageEvent.BACKGROUNDIMAGE_LAST = 4;

public final static int org.havi.ui.event.HEventRepresentation.ER_TYPE_NOT_SUPPORTED = 0;
public final static int org.havi.ui.event.HEventRepresentation.ER_TYPE_STRING = 1;
public final static int org.havi.ui.event.HEventRepresentation.ER_TYPE_COLOR = 2;
public final static int org.havi.ui.event.HEventRepresentation.ER_TYPE_SYMBOL = 4;

public final static int org.havi.ui.event.HFocusEvent.HFOCUS_FIRST = 2025;
public final static int org.havi.ui.event.HFocusEvent.FOCUS_TRANSFER = 2025;
public final static int org.havi.ui.event.HFocusEvent.HFOCUS_LAST = 2025;
public final static int org.havi.ui.event.HFocusEvent.NO_TRANSFER_ID = -1;

public final static int org.havi.ui.event.HItemEvent.ITEM_FIRST = 2006;
public final static int org.havi.ui.event.HItemEvent.ITEM_START_CHANGE = 2006;
public final static int org.havi.ui.event.HItemEvent.ITEM_TOGGLE_SELECTED = 2007;
public final static int org.havi.ui.event.HItemEvent.ITEM_SELECTED = 2008;
public final static int org.havi.ui.event.HItemEvent.ITEM_CLEARED = 2009;
public final static int org.havi.ui.event.HItemEvent.ITEM_SELECTION_CLEARED = 2010;
public final static int org.havi.ui.event.HItemEvent.ITEM_SET_CURRENT = 2011;
public final static int org.havi.ui.event.HItemEvent.ITEM_SET_PREVIOUS = 2012;
public final static int org.havi.ui.event.HItemEvent.ITEM_SET_NEXT = 2013;
public final static int org.havi.ui.event.HItemEvent.ITEM_END_CHANGE = 2014;
public final static int org.havi.ui.event.HItemEvent.ITEM_LAST = 2014;

public final static int org.havi.ui.event.HRcEvent.RC_FIRST = 400;
public final static int org.havi.ui.event.HRcEvent.VK_COLORED_KEY_0 = 403;
public final static int org.havi.ui.event.HRcEvent.VK_COLORED_KEY_1 = 404;
public final static int org.havi.ui.event.HRcEvent.VK_COLORED_KEY_2 = 405;
public final static int org.havi.ui.event.HRcEvent.VK_COLORED_KEY_3 = 406;
public final static int org.havi.ui.event.HRcEvent.VK_COLORED_KEY_4 = 407;
public final static int org.havi.ui.event.HRcEvent.VK_COLORED_KEY_5 = 408;
public final static int org.havi.ui.event.HRcEvent.VK_POWER = 409;
public final static int org.havi.ui.event.HRcEvent.VK_DIMMER = 410;
public final static int org.havi.ui.event.HRcEvent.VK_WINK = 411;
public final static int org.havi.ui.event.HRcEvent.VK_REWIND = 412;
public final static int org.havi.ui.event.HRcEvent.VK_STOP = 413;
public final static int org.havi.ui.event.HRcEvent.VK_EJECT_TOGGLE = 414;
public final static int org.havi.ui.event.HRcEvent.VK_PLAY = 415;
public final static int org.havi.ui.event.HRcEvent.VK_RECORD = 416;
public final static int org.havi.ui.event.HRcEvent.VK_FAST_FWD = 417;
public final static int org.havi.ui.event.HRcEvent.VK_PLAY_SPEED_UP = 418;
public final static int org.havi.ui.event.HRcEvent.VK_PLAY_SPEED_DOWN = 419;
public final static int org.havi.ui.event.HRcEvent.VK_PLAY_SPEED_RESET = 420;
public final static int org.havi.ui.event.HRcEvent.VK_RECORD_SPEED_NEXT = 421;
public final static int org.havi.ui.event.HRcEvent.VK_GO_TO_START = 422;
public final static int org.havi.ui.event.HRcEvent.VK_GO_TO_END = 423;
public final static int org.havi.ui.event.HRcEvent.VK_TRACK_PREV = 424;
public final static int org.havi.ui.event.HRcEvent.VK_TRACK_NEXT = 425;
public final static int org.havi.ui.event.HRcEvent.VK_RANDOM_TOGGLE = 426;
public final static int org.havi.ui.event.HRcEvent.VK_CHANNEL_UP = 427;
public final static int org.havi.ui.event.HRcEvent.VK_CHANNEL_DOWN = 428;
public final static int org.havi.ui.event.HRcEvent.VK_STORE_FAVORITE_0 = 429;
public final static int org.havi.ui.event.HRcEvent.VK_STORE_FAVORITE_1 = 430;
public final static int org.havi.ui.event.HRcEvent.VK_STORE_FAVORITE_2 = 431;
public final static int org.havi.ui.event.HRcEvent.VK_STORE_FAVORITE_3 = 432;
public final static int org.havi.ui.event.HRcEvent.VK_RECALL_FAVORITE_0 = 433;
public final static int org.havi.ui.event.HRcEvent.VK_RECALL_FAVORITE_1 = 434;
public final static int org.havi.ui.event.HRcEvent.VK_RECALL_FAVORITE_2 = 435;
public final static int org.havi.ui.event.HRcEvent.VK_RECALL_FAVORITE_3 = 436;
public final static int org.havi.ui.event.HRcEvent.VK_CLEAR_FAVORITE_0 = 437;
public final static int org.havi.ui.event.HRcEvent.VK_CLEAR_FAVORITE_1 = 438;
public final static int org.havi.ui.event.HRcEvent.VK_CLEAR_FAVORITE_2 = 439;
public final static int org.havi.ui.event.HRcEvent.VK_CLEAR_FAVORITE_3 = 440;

```

```
public final static int org.havi.ui.event.HRcEvent.VK_SCAN_CHANNELS_TOGGLE = 441;
public final static int org.havi.ui.event.HRcEvent.VK_PINP_TOGGLE = 442;
public final static int org.havi.ui.event.HRcEvent.VK_SPLIT_SCREEN_TOGGLE = 443;
public final static int org.havi.ui.event.HRcEvent.VK_DISPLAY_SWAP = 444;
public final static int org.havi.ui.event.HRcEvent.VK_SCREEN_MODE_NEXT = 445;
public final static int org.havi.ui.event.HRcEvent.VK_VIDEO_MODE_NEXT = 446;
public final static int org.havi.ui.event.HRcEvent.VK_VOLUME_UP = 447;
public final static int org.havi.ui.event.HRcEvent.VK_VOLUME_DOWN = 448;
public final static int org.havi.ui.event.HRcEvent.VK_MUTE = 449;
public final static int org.havi.ui.event.HRcEvent.VK_SURROUND_MODE_NEXT = 450;
public final static int org.havi.ui.event.HRcEvent.VK_BALANCE_RIGHT = 451;
public final static int org.havi.ui.event.HRcEvent.VK_BALANCE_LEFT = 452;
public final static int org.havi.ui.event.HRcEvent.VK_FADER_FRONT = 453;
public final static int org.havi.ui.event.HRcEvent.VK_FADER_REAR = 454;
public final static int org.havi.ui.event.HRcEvent.VK_BASS_BOOST_UP = 455;
public final static int org.havi.ui.event.HRcEvent.VK_BASS_BOOST_DOWN = 456;
public final static int org.havi.ui.event.HRcEvent.VK_INFO = 457;
public final static int org.havi.ui.event.HRcEvent.VK_GUIDE = 458;
public final static int org.havi.ui.event.HRcEvent.VK_TELETEXT = 459;
public final static int org.havi.ui.event.HRcEvent.VK_SUBTITLE = 460;
public final static int org.havi.ui.event.HRcEvent.RC_LAST = 460;
public final static int org.havi.ui.event.HTextEvent.TEXT_FIRST = 2015;
public final static int org.havi.ui.event.HTextEvent.TEXT_LAST = 2024;
public final static int org.havi.ui.event.HTextEvent.TEXT_START_CHANGE = 2015;
public final static int org.havi.ui.event.HTextEvent.TEXT_CHANGE = 2016;
public final static int org.havi.ui.event.HTextEvent.TEXT_CARET_CHANGE = 2017;
public final static int org.havi.ui.event.HTextEvent.TEXT_END_CHANGE = 2018;
public final static int org.havi.ui.event.HTextEvent.CARET_NEXT_CHAR = 2019;
public final static int org.havi.ui.event.HTextEvent.CARET_NEXT_LINE = 2020;
public final static int org.havi.ui.event.HTextEvent.CARET_PREV_CHAR = 2021;
public final static int org.havi.ui.event.HTextEvent.CARET_PREV_LINE = 2022;
public final static int org.havi.ui.event.HTextEvent.CARET_NEXT_PAGE = 2023;
public final static int org.havi.ui.event.HTextEvent.CARET_PREV_PAGE = 2024;
```

# Package org.havi.ui

Class Summary	
<b>Interfaces</b>	
<code>HActionable</code>	This interface is implemented by all HAVi UI components that can be actioned by the user.
<code>HActionInputPreferred</code>	A component which implements <code>HActionInputPreferred</code> indicates that this component expects to receive <code>HActionEvent</code> input events.
<code>HAdjustableLook</code>	The <code>HAdjustableLook</code> interface is implemented by all platform looks which support adjustable components (i.e.
<code>HAdjustmentInputPreferred</code>	A component which implements <code>HAdjustmentInputPreferred</code> indicates that this component expects to receive <code>HAdjustmentEvent</code> input events.
<code>HAdjustmentValue</code>	This interface is implemented by all HAVi UI components which have some form of adjustable numerical value (e.g.
<code>HAnimateEffect</code>	The <code>HAnimateEffect</code> interface defines effect constants and controls for time-varying animations.
<code>HComponentOrdering</code>	This interface is implemented for all HAVi component containers that support the manipulation of the z-ordering of their children.
<code>HItemValue</code>	This interface is implemented by all HAVi UI components which have some form of selectable content (e.g.
<code>HKeyboardInputPreferred</code>	A component which implements <code>HKeyboardInputPreferred</code> indicates that this component expects to receive both <code>HKeyEvent</code> and <code>HTextEvent</code> input events.
<code>HLook</code>	The <code>HLook</code> interface defines the "look" of a component and may be regarded as a mechanism to allow a "pluggable" paint method to be attached to the component.
<code>HMatte</code>	<code>HMatte</code> is the base interface for all matte classes.
<code>HMatteLayer</code>	This <code>HMatteLayer</code> interface enables the presentation of components, together with an associated <code>HMatte</code> , for matte compositing.
<code>HNavigable</code>	This interface is implemented by all HAVi UI components that can be navigated to by the user (i.e.
<code>HNavigationInputPreferred</code>	A component which implements <code>HNavigationInputPreferred</code> indicates that this component expects to receive <code>HFocusEvent</code> events.
<code>HNoInputPreferred</code>	A component which implements <code>HNoInputPreferred</code> indicates that the user cannot navigate to this component.
<code>HOrientable</code>	The <code>HOrientable</code> interface provides support for components which have an orientation.
<code>HSelectionInputPreferred</code>	A component which implements <code>HSelectionInputPreferred</code> indicates that this component expects to receive <code>HItemEvent</code> input events.
<code>HState</code>	The <code>HState</code> interface encapsulates constants for component states which are used in the various <code>HVisible</code> <code>setContent</code> and <code>getContent</code> methods, to indicate which state the specified content is to be set.
<code>HSwitchable</code>	This interface is implemented for all user interface components that can be actioned such that they "toggle" on and off and maintain the chosen state.

Class Summary	
<code>HTextLayoutManager</code>	The <code>HTextLayoutManager</code> class manages the layout and rendering on-screen of a "marked-up" string.
<code>HTextValue</code>	This interface is implemented by all HAVi UI components which have have editable text content (e.g.
<code>HVersion</code>	The <code>HVersion</code> interface defines some versioning constants that are accessible by using the <code>java.lang.System</code> method <code>getProperty</code> , with the appropriate property name.
<b>Classes</b>	
<code>HAnimateLook</code>	The <code>HAnimateLook</code> class is used by any <code>HVisible</code> component to display animated graphical content.
<code>HAnimation</code>	The <code>HAnimation</code> class is a user interface component used to display animated graphical content (as <code>HStaticAnimation</code> ) which additionally enables a user to navigate to it, i.e.
<code>HBackgroundConfigTemplate</code>	The <code>HBackgroundConfigTemplate</code> class is used to obtain a valid <code>HBackgroundConfiguration</code> .
<code>HBackgroundConfiguration</code>	The <code>HBackgroundConfiguration</code> class describes the characteristics (settings) of an <code>HBackgroundDevice</code> .
<code>HBackgroundDevice</code>	This class represents the ultimate background of a screen.
<code>HBackgroundImage</code>	This class represents a background image.
<code>HChangeData</code>	A class used as argument for <code>widgetChanged(HVisible, HChangeData[])</code> .
<code>HComponent</code>	The <code>HComponent</code> class extends the <code>java.awt.Component</code> class by implementing the <code>HMatteLayer</code> interface.
<code>HContainer</code>	The <code>HContainer</code> class extends the <code>java.awt.Container</code> class by implementing the <code>HMatteLayer</code> interface and providing additional Z-ordering capabilities, which are required since components in the HAVi user-interface are explicitly allowed to overlap each other.
<code>HDefaultTextLayoutManager</code>	The <code>HDefaultTextLayoutManager</code> provides the default text rendering mechanism for the <code>HStaticText</code> <code>HText</code> and <code>HTextButton</code> classes.
<code>HEmulatedGraphicsConfiguration</code>	An <code>HEmulatedGraphicsConfiguration</code> is a configuration for a "virtual" graphics device that may perform one or more emulations, e.g.
<code>HEmulatedGraphicsDevice</code>	An <code>HEmulatedGraphicsDevice</code> is a "virtual" graphics device that has the capability to be configured to perform one (of many) possible emulations.
<code>HEventMulticaster</code>	The <code>HEventMulticaster</code> class is intended to handle event dispatching for the following HAVi events: <code>HBackgroundImageEvent</code> <code>HScreenConfigurationEvent</code> <code>HScreenLocationModifiedEvent</code> <code>HActionEvent</code> <code>HFocusEvent</code> <code>HItemEvent</code> <code>HTextEvent</code> <code>HKeyEvent</code> <code>HAdjustmentEvent</code> <code>java.awt.event.WindowEvent</code> <code>org.davic.resources.ResourceStatusEvent</code>
<code>HFlatEffectMatte</code>	The <code>HFlatEffectMatte</code> class represents a matte that is constant over space but varies over time.



Class Summary	
<code>HFlatMatte</code>	The <code>HFlatMatte</code> class represents a matte that is constant over space and time.
<code>HFontCapabilities</code>	The <code>HFontCapabilities</code> class allows applications to query the rendering support for various character ranges and individual characters within specified fonts.
<code>HGraphicButton</code>	The <code>HGraphicButton</code> is a user interface component representing a push-button with static read-only graphical content.
<code>HGraphicLook</code>	The <code>HGraphicLook</code> class is used by any <code>HVisible</code> component to display static graphical content.
<code>HGraphicsConfigTemplate</code>	The <code>HGraphicsConfigTemplate</code> class is used to obtain a valid <code>HGraphicsConfiguration</code> .
<code>HGraphicsConfiguration</code>	The <code>HGraphicsConfiguration</code> class describes the characteristics (settings) of an <code>HGraphicsDevice</code> .
<code>HGraphicsDevice</code>	The <code>HGraphicsDevice</code> class describes the raster graphics devices that are available for a particular <code>HScreen</code> .
<code>HIcon</code>	The <code>HIcon</code> is a user interface component used to display static graphical content (as <code>HStaticIcon</code> ) which also enables a user to navigate to it, i.e.
<code>HImageEffectMatte</code>	The <code>HImageEffectMatte</code> class represents a matte that varies over both space and time, it is specified as a sequence of image masks.
<code>HImageHints</code>	The <code>HImageHints</code> object allows an application to pass hints to the system how best to tailor an image to match a (possibly) restricted <code>HGraphicsConfiguration</code> .
<code>HImageMatte</code>	The <code>HImageMatte</code> class represents a matte that varies over space but is constant over time, it can be specified by an "image mask" (a single channel image) where the pixels indicate matte transparency.
<code>HListElement</code>	<code>HListElement</code> is a holder for content used with the <code>HListGroup</code> component.
<code>HListGroup</code>	The <code>HListGroup</code> is a user interface component representing a list of selectable items ( <code>HListElement</code> ) which contain static read-only graphical and / or textual content.
<code>HListGroupLook</code>	The <code>HListGroupLook</code> class is used by the <code>HListGroup</code> component to display both the <code>HListGroup</code> itself (potentially including a scrollbar component) and graphical or textual list content held on the <code>HListGroup</code> .
<code>HMultilineEntry</code>	The <code>HMultilineEntry</code> is a user interface component used to receive multiple lines of alphanumeric entry from the user.
<code>HMultilineEntryLook</code>	The <code>HMultilineEntryLook</code> class is used by the <code>HMultilineEntry</code> component to display the entering of text.
<code>HRange</code>	The <code>HRange</code> is a user interface component used to display a value within a fixed range (as <code>org.havi.ui.HStaticRange HStaticRange</code> ) which also enables a user to navigate to it, i.e.
<code>HRangeLook</code>	The <code>HRangeLook</code> class is used to display a slider or scrollbar type range control on screen.
<code>HRangeValue</code>	The <code>HRangeValue</code> is a user interface component used to display a value within a fixed range (as <code>org.havi.ui.HStaticRange HStaticRange</code> ) which enables a user to navigate to and alter the value of it, i.e.
<code>HScene</code>	An <code>HScene</code> is a container representing the displayable area on-screen within which the application can display itself and thus interact with the user.
<code>HSceneFactory</code>	The <code>HSceneFactory</code> class provides a generic mechanism for an application to request <code>HScene</code> resources from a (conceptual) window management system.

Class Summary	
HSceneTemplate	The <code>HSceneTemplate</code> class is used to obtain an <code>HScene</code> subject to a variety of constraints.
HScreen	This class describes the final output composition of a device.
HScreenConfigTemplate	This class describes a configuration of a screen device in terms of various properties and their importance to the application.
HScreenConfiguration	The <code>HScreenConfiguration</code> class describes the characteristics (settings) of an <code>HScreenDevice</code> .
HScreenDevice	An instance of the <code>HScreen</code> class represents a single independent video output signal from a device.
HScreenDimension	<code>HScreenDimension</code> denotes a screen dimension expressed as a relative value of the screen dimensions.
HScreenPoint	<code>HScreenPoint</code> denotes a screen location expressed as a relative value of the screen dimensions.
HScreenRectangle	<code>HScreenRectangle</code> denotes a screen area expressed as a relative value of the screen dimensions.
HSinglelineEntry	The <code>HSinglelineEntry</code> is a user interface component used to receive a single line of alphanumeric entry from the user and can also be used for password input.
HSinglelineEntryLook	The <code>HSinglelineEntryLook</code> class is used by the <code>HSinglelineEntry</code> component to display the entering of text.
HSound	The <code>HSound</code> class is used to represent an audio clip.
HStaticAnimation	The <code>HStaticAnimation</code> is a user interface component used to display animated graphical content but which does <i>not</i> permit the user to navigate (focus) upon it.
HStaticIcon	The <code>HStaticIcon</code> is a user interface component used to display static graphical content which does <i>not</i> permit the user to navigate (focus) upon it.
HStaticRange	The <code>HStaticRange</code> is a user interface component used to display a static value which is within a fixed range, but does <i>not</i> permit the user to navigate (focus) upon it.
HStaticText	The <code>HStaticText</code> is a user interface component used to display static read-only textual content which does <i>not</i> permit the user to navigate (focus) upon it.
HStillImageBackground-Configuration	This class represents a background configuration which supports the installation of still images.
HText	The <code>HText</code> is a user interface component used to display static read-only textual content (as <code>HStaticText</code> ) which additionally enables a user to navigate to it, i.e.
HTextButton	The <code>HTextButton</code> is a user interface component representing a push-button with static read-only textual content.
HTextLook	The <code>HTextLook</code> class is used by any <code>HVisible</code> component to display static text content.
HToggleButton	The <code>HToggleButton</code> is a user interface component representing a "check box", or with the support of the <code>HToggleGroup</code> class, "radio buttons".
HToggleGroup	<code>HToggleButton</code> within the same <code>HToggleGroup</code> will behave so that a maximum of one <code>HToggleButton</code> has switchable state true, as returned by <code>getSwitchableState()</code> , so as to achieve a "radio button" effect.
HVideoComponent	<code>HVideoComponent</code> is an opaque class encapsulating the presentation of a video source <i>within</i> an application, i.e.

<b>Class Summary</b>	
<code>HVideoConfigTemplate</code>	The <code>HVideoConfigTemplate</code> class is used to obtain a valid <code>HVideoConfiguration</code> .
<code>HVideoConfiguration</code>	The <code>HVideoConfiguration</code> class describes the characteristics (settings) of an <code>HVideoDevice</code> .
<code>HVideoDevice</code>	The <code>HVideoDevice</code> class describes the logical video devices which can contribute to the appearance of a particular screen.
<code>HVisible</code>	The <code>HVisible</code> class is the base class for all non-interactive components.
<b>Exceptions</b>	
<code>HConfigurationException</code>	Thrown when an application requests an <code>HScreenConfiguration</code> that cannot be satisfied -- either because the <code>HScreenConfiguration</code> does not have the functionality, or because the requested <code>HScreenConfiguration</code> is otherwise invalid, e.g.
<code>HInvalidLookException</code>	An <code>HInvalidLookException</code> is an exception that is thrown when a particular <code>HLook</code> is not compatible with the <code>HVisible</code> component it has been associated with.
<code>HMatteException</code>	An <code>HMatteException</code> is an exception that is thrown when a Component is unable to support the desired <code>HMatte</code> effect.
<code>HPermissionDeniedException</code>	Thrown when an application attempts to control a device or feature it does not have the right to do at the time of making the call.
<code>HUIException</code>	<code>HUIException</code> is a generic exception that indicates that the desired user-interface mechanism cannot be performed for some reason.

# org.havi.ui

# HActionable

## Syntax

`public interface HActionable extends HNavigable, HActionInputPreferred`

## All Known Subinterfaces:

`HSwitchable`

## All Superinterfaces:

`HActionInputPreferred, HNavigable, HNavigationInputPreferred`

## All Known Implementing Classes:

`HTextButton, HGraphicButton`

## Description

This interface is implemented by all HAVi UI components that can be actioned by the user.

Event Behaviour

Subclasses of `HComponent` which implement `HActionable` must respond to `HFocusEvent` and `HActionEvent` events.

Applications should assume that classes which implement `HActionable` can generate events of the types `HFocusEvent` and `HActionEvent` in response to other types of input event.

An application may add one or more `HActionListener` listeners to the component. The `actionPerformed` method of the `HActionListener` is invoked whenever the `HActionable` is actioned.

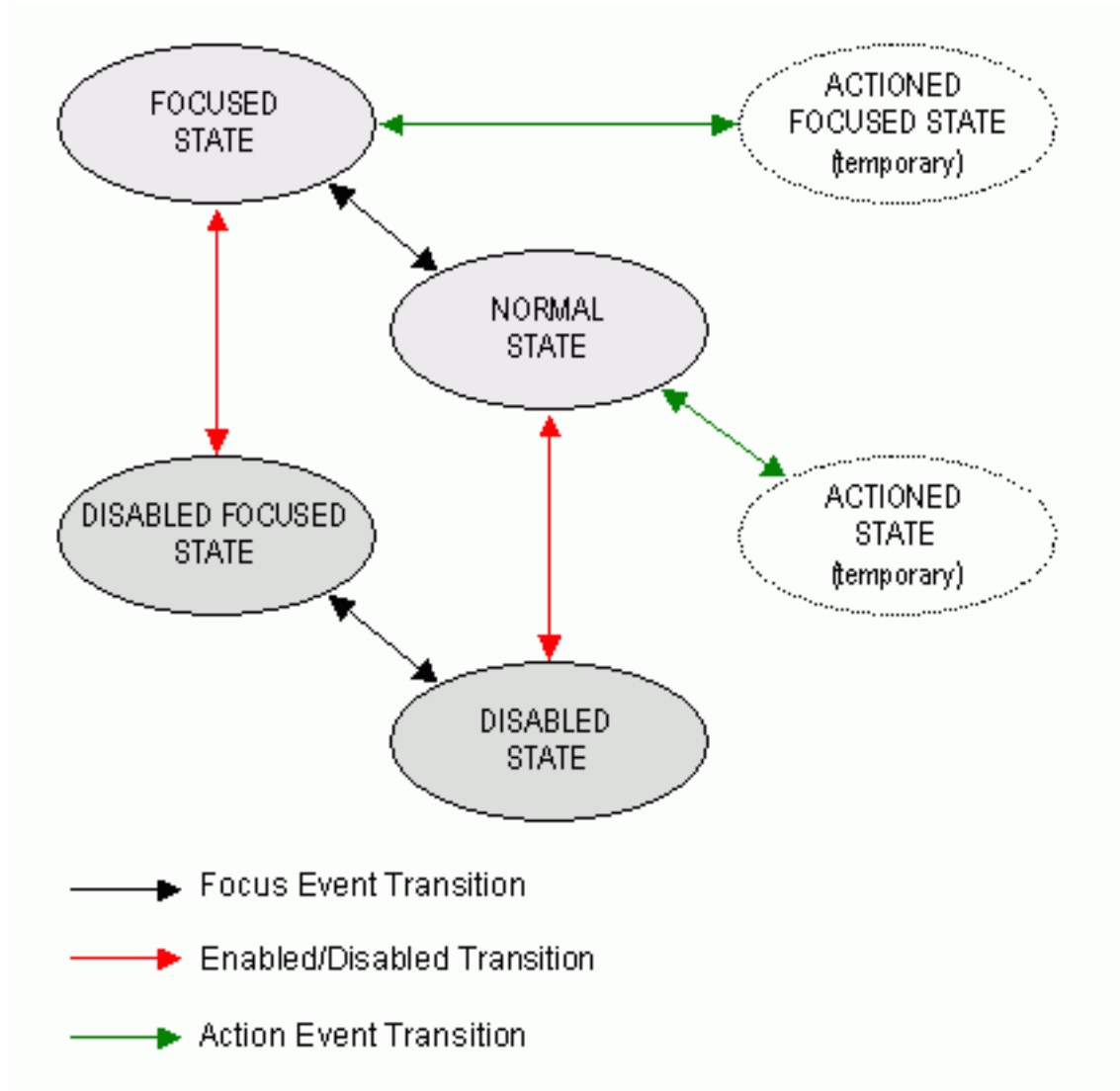
HAVi action events are discussed in detail in the `HActionInputPreferred` interface description.

Interaction States

The following interaction states are valid for this `HActionable` component:

- `NORMAL_STATE`
- `FOCUSED_STATE`
- `ACTIONED_STATE`
- `ACTIONED_FOCUSED_STATE`
- `DISABLED_STATE`
- `DISABLED_FOCUSED_STATE`

The state machine diagram below shows the valid state transitions for an `HActionable` component.



Unlike `HSwitchable` components the transition back from an actioned state (i.e. one with the `ACTIONED_STATE_BIT` bit set is automatically fired once all registered `HActionListener` listeners have been called.

A direct consequence of is that `HActionable` components can only achieve the `ACTIONED_STATE` and `ACTIONED_FOCUSED_STATE` states on a temporary basis.

`HActionable` components may not be disabled while actioned.

#### Platform Classes

The following HAVi platform classes implement or inherit the `HActionable` interface. These classes shall all generate both `HFocusEvent` and `HActionEvent` events in addition to any other events specified in the respective class descriptions.

- `HGraphicButton`
- `HTextButton`
- `HToggleButton`
- `HListGroup`

**See Also:**

`HNavigable`, `HActionInputPreferred`, `HActionEvent`, `HActionListener`

## Methods

---

**addActionListener(HActionListener)**

```
public void addActionListener(HActionListener l)
```

Adds the specified `HActionListener` to receive `HActionEvent` events sent from this `HActionable`. If the listener has already been added further calls will add further references to the listener, which will then receive multiple copies of a single event.

**Parameters:**

l - the `HActionListener`.

---

**getActionCommand()**

```
public java.lang.String getActionCommand()
```

Gets the command name for the `HActionEvent` event fired by this `HActionable`.

**Returns:**

A `String` representing the command name of the `HActionEvent` fired by this `HActionable`.

**See Also:**

`getActionCommand()`

---

**getActionSound()**

```
public HSound getActionSound()
```

Associate a sound to be played when the interaction state of the `HActionable` makes the following transitions:

- `NORMAL_STATE` to `ACTIONED_STATE`
- `FOCUSED_STATE` to `ACTIONED_FOCUSED_STATE`

**Returns:**

The sound played when the component is actioned.

---

**removeActionListener(HActionListener)**

```
public void removeActionListener(HActionListener l)
```

Removes the specified `HActionListener` so that it no longer receives `HActionEvent` events from this `HActionable`. If the specified listener is not registered, the method has no effect. If multiple references to a single listener have been registered it should be noted that this method will only remove one reference per call.

**Parameters:**

l - the `HActionListener`.

---

**setActionCommand(String)**

```
public void setActionCommand(java.lang.String command)
```

Sets the command name for the `HActionEvent` event fired by this `HActionable`.

**Parameters:**

`command` - a `String` used to set the action command.

**See Also:**

`getActionCommand()`

---

**setActionSound(HSound)**

```
public void setActionSound(HSound sound)
```

Associate a sound to be played when the interaction state of the `HActionable` makes the following transitions:

- `NORMAL_STATE` to `ACTIONED_STATE`
- `FOCUSED_STATE` to `ACTIONED_FOCUSED_STATE`

**Parameters:**

`sound` - the sound to be played, when the component is actioned. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

# org.havi.ui

# HActionInputPreferred

## Syntax

```
public interface HActionInputPreferred
```

## All Known Subinterfaces:

```
HActionable, HSwitchable
```

## Description

A component which implements `HActionInputPreferred` indicates that this component expects to receive `HActionEvent` input events.

All interoperable implementations of the `HActionInputPreferred` interface must extend `HComponent`.

Note that the `java.awt.Component` method `isFocusTraversable` should always return true for a `java.awt.Component` implementing this interface.

## Methods

---

### `processHActionEvent(HActionEvent)`

```
public void processHActionEvent(HActionEvent evt)
```

Process an `HActionEvent` sent to this `HActionInputPreferred`.

#### Parameters:

`evt` - the `HActionEvent` to process.



# org.havi.ui

## HAdjustableLook

### Syntax

```
public interface HAdjustableLook extends HLook
```

### All Superinterfaces:

java.lang.Cloneable, HLook

### All Known Implementing Classes:

HRangeLook, HListGroupLook

### Description

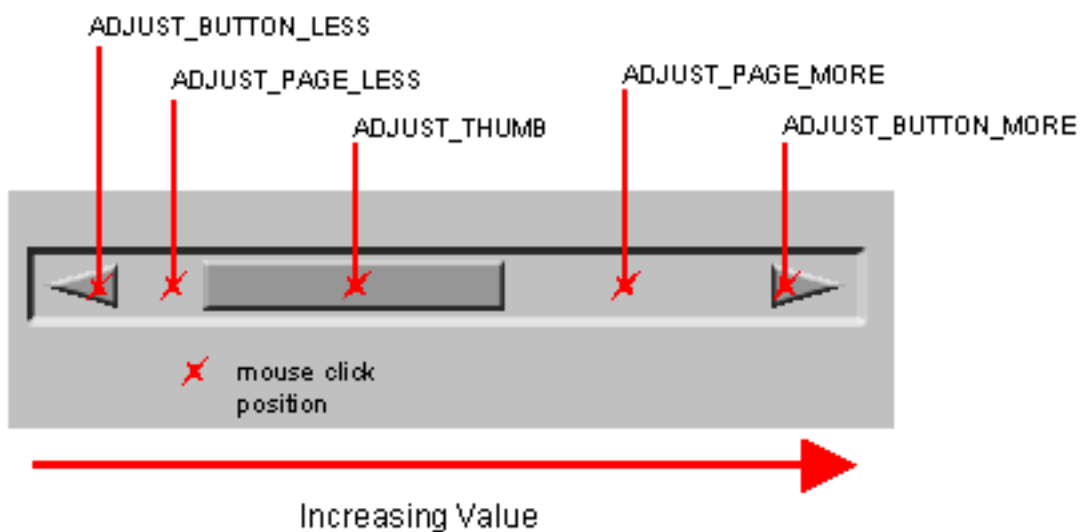
The `HAdjustableLook` interface is implemented by all platform looks which support adjustable components (i.e. those components which implement the `HAdjustmentValue` interface. The following platform looks shall implement this interface:

- `HRangeLook`
- `HListGroupLook`

The `HAdjustableLook` interface supports pointer based systems by providing a mechanism of "hit-testing" which allows the `HAdjustmentValue` component to determine which part of the on-screen representation has been clicked in, and to adjust its internal value accordingly.

The diagram below shows one possible on-screen representation of an `HAdjustmentValue` component, with `ORIENT_LEFT_TO_RIGHT` orientation.

### Example hitTest() return values (for ORIENT\_LEFT\_TO\_RIGHT orientation)



HLook implementations which implement `HAdjustableLook` may use the `getOrientation()` method to determine the appropriate constant to return from `hitTest(HAdjustmentValue, Point)` since the correct constant is dependent on the orientation of the component.

It is a valid implementation option to return `ADJUST_NONE` from the `hitTest(HAdjustmentValue, Point)` method in all cases.

It is a valid implementation option to never return `ADJUST_BUTTON_LESS` and `ADJUST_BUTTON_MORE` in the case where such active areas are not presented on screen by the `HLook`.

## Fields

---

### ADJUST\_BUTTON\_LESS

```
public static final int ADJUST_BUTTON_LESS
```

A constant which may be returned from the `hitTest(HAdjustmentValue, Point)` method to indicate that the pointer was clicked in an adjustment area which indicates that the adjustable value should be decremented by one unit.

Such an area should be drawn with an arrow pointing towards the minimum end of the range, according to the orientation as retrieved with `getOrientation()`.

Use of this constant is implementation-specific.

---

### ADJUST\_BUTTON\_MORE

```
public static final int ADJUST_BUTTON_MORE
```

A constant which may be returned from the `hitTest(HAdjustmentValue, Point)` method to indicate that the pointer was clicked in an adjustment area which indicates that the adjustable value should be incremented by one unit.

Such an area should be drawn with an arrow pointing towards the maximum end of the range, according to the orientation as retrieved with `getOrientation()`.

Use of this constant is implementation-specific.

---

### ADJUST\_NONE

```
public static final int ADJUST_NONE
```

A constant which may be returned from the `hitTest(HAdjustmentValue, Point)` method to indicate that the pointer was not clicked over an active adjustment area.

---

### ADJUST\_PAGE\_LESS

```
public static final int ADJUST_PAGE_LESS
```

A constant which may be returned from the `hitTest(HAdjustmentValue, Point)` method to indicate that the pointer was clicked in an adjustment area which indicates that the adjustable value should be decremented by one block.

---

### ADJUST\_PAGE\_MORE

```
public static final int ADJUST_PAGE_MORE
```

A constant which may be returned from the `hitTest(HAdjustmentValue, Point)` method to indicate that the pointer was clicked in an adjustment area which indicates that the adjustable value should be incremented by one block.

---

## ADJUST\_THUMB

```
public static final int ADJUST_THUMB
```

A constant which may be returned from the `hitTest(HAdjustmentValue, Point)` method to indicate that the pointer was clicked in an adjustment area which indicates that the adjustable value should change according to pointer motion events received by the component, until the pointer button is released.

## Methods

---

### hitTest(HAdjustmentValue, Point)

```
public int hitTest(HAdjustmentValue component, java.awt.Point pt)
```

Returns a value which indicates the pointer click position in the on-screen representation of the adjustable component. Note that it is a valid implementation option to always return `ADJUST_NONE`.

**Parameters:**

`component` - the `HAdjustmentValue` component for which the hit position should be calculated.

`pt` - the pointer click point.

**Returns:**

one of `ADJUST_NONE`, `ADJUST_BUTTON_LESS`, `ADJUST_PAGE_LESS`, `ADJUST_THUMB`, `ADJUST_PAGE_MORE` or `ADJUST_BUTTON_MORE`.

# org.havi.ui HAdjustmentInputPreferred

## Syntax

```
public interface HAdjustmentInputPreferred extends HOrientable
```

## All Known Subinterfaces:

```
HAdjustmentValue
```

## All Superinterfaces:

```
HOrientable
```

## All Known Implementing Classes:

```
HRangeValue
```

## Description

A component which implements `HAdjustmentInputPreferred` indicates that this component expects to receive `HAdjustmentEvent` input events.

The system must provide a means of generating `HAdjustmentEvent` events as necessary. For platforms with a restricted number of physical keys this may involve a "virtual keyboard" or similar mechanism.

All interoperable implementations of the `HAdjustmentInputPreferred` interface must extend `HComponent`.

Note that the `java.awt.Component` method `isFocusTraversable` should always return true for a `java.awt.Component` implementing this interface.

## Methods

---

### `getAdjustMode()`

```
public boolean getAdjustMode()
```

Get the adjustment mode for this `HAdjustmentInputPreferred`. If the returned value is `true` the component is in adjustment mode, and its value may be changed on receipt of `ADJUST_LESS` and `ADJUST_MORE` events.

The component is switched into and out of adjustment mode on receiving `ADJUST_START_CHANGE` and `ADJUST_END_CHANGE` events.

#### Returns:

`true` if this component is in adjustment mode, false otherwise.

---

### `processHAdjustmentEvent(HAdjustmentEvent)`

```
public void processHAdjustmentEvent(HAdjustmentEvent evt)
```

Process an `HAdjustmentEvent` sent to this `HAdjustmentInputPreferred`.

#### Parameters:

`evt` - the `HAdjustmentEvent` to process.

---

**setAdjustMode(boolean)**

```
public void setAdjustMode(boolean adjust)
```

Set the adjustment mode for this `HAdjustmentInputPreferred`.

This method is provided for the convenience of component implementors. Interoperable applications shall not call this method. It cannot be made protected because interfaces cannot have protected methods.

**Parameters:**

`edit` - true to switch this component into adjustment mode, false otherwise.

**See Also:**

`getAdjustMode()`

# org.havi.ui

# HAdjustmentValue

## Syntax

```
public interface HAdjustmentValue extends HNavigable, HAdjustmentInputPreferred
```

## All Superinterfaces:

```
HAdjustmentInputPreferred, HNavigable, HNavigationInputPreferred,  
HOrientable
```

## All Known Implementing Classes:

```
HRangeValue
```

## Description

This interface is implemented by all HAVi UI components which have some form of adjustable numerical value (e.g. a range control)

### Event Behavior

Subclasses of `HComponent` which implement `HAdjustmentValue` must respond to `HFocusEvent` and `HAdjustmentEvent` events.

Applications should assume that classes which implement `HAdjustmentValue` can generate events of the types `HFocusEvent` and `HAdjustmentEvent` in response to other types of input event.

An application may add one or more `HAdjustmentListener` listeners to the component. The `valueChanged(HAdjustmentEvent)` method of the `HAdjustmentListener` is invoked whenever the value of the `HAdjustmentValue` is modified.

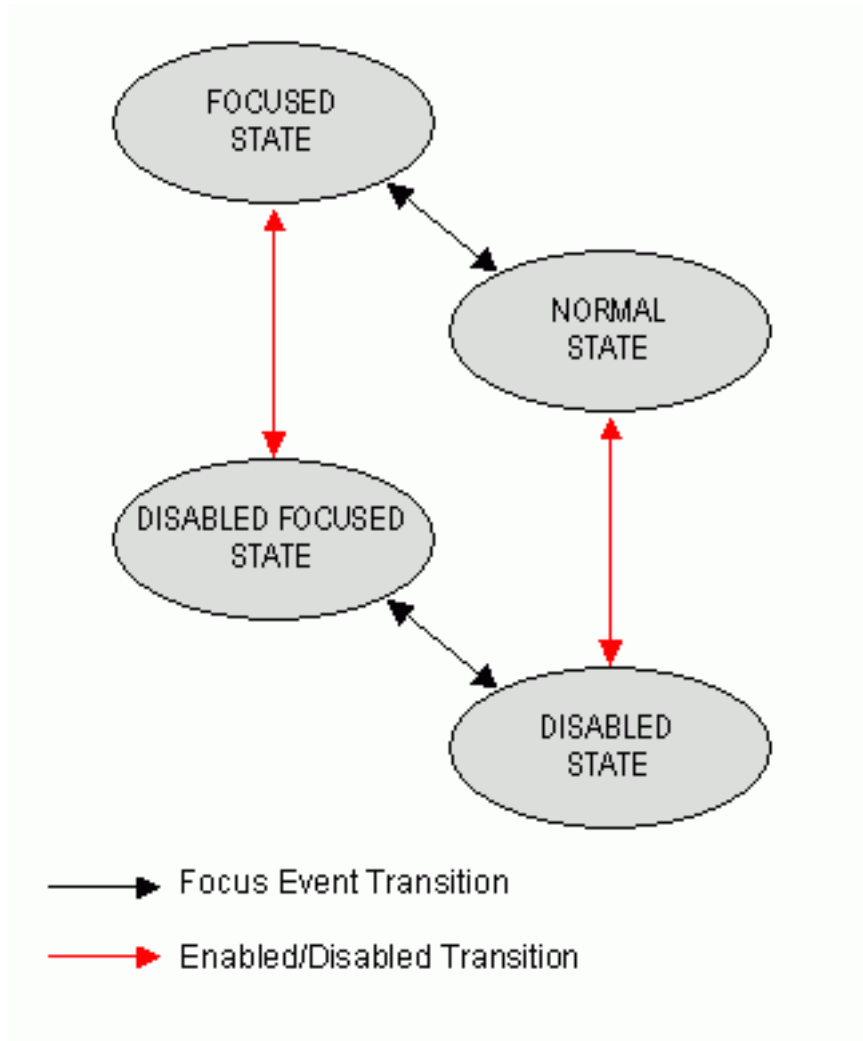
HAVi adjustment events are discussed in detail in the `HAdjustmentInputPreferred` interface description.

### Interaction States

The following interaction states are valid for this `HAdjustmentValue` component:

- `NORMAL_STATE`
- `FOCUSED_STATE`
- `DISABLED_STATE`
- `DISABLED_FOCUSED_STATE`

The state machine diagram below shows the valid state transitions for an `HAdjustmentValue` component.



### Platform Classes

The following HAVi platform classes implement or inherit the `HAdjustmentValue` interface. These classes shall all generate both `HFocusEvent` and `HAdjustmentEvent` events in addition to any other events specified in the respective class descriptions.

- `HRangeValue`
- `HListGroup`

### See Also:

`HNavigable`, `HOrientable`, `HAdjustmentInputPreferred`, `HAdjustmentEvent`, `HAdjustmentListener`

## Methods

---

### **addAdjustmentListener(HAdjustmentListener)**

```
public void addAdjustmentListener(HAdjustmentListener l)
```

Adds the specified `HAdjustmentListener` to receive `HAdjustmentEvent` sent from this object. If the listener has already been added further calls will add further references to the listener, which will then receive multiple copies of a single event.

**Parameters:**

1 - the `HAdjustmentListener` to be notified.

---

**getAdjustmentSound()**

```
public HSound getAdjustmentSound()
```

Get the sound to be played when the value changes.

**Returns:**

The sound played when the value changes

---

**getBlockIncrement()**

```
public int getBlockIncrement()
```

Get the block increment for this `HAdjustmentValue`.

**Returns:**

the block increment value for this `HAdjustmentValue`.

---

**getUnitIncrement()**

```
public int getUnitIncrement()
```

Get the unit increment for this `HAdjustmentValue`.

**Returns:**

the increment value for this `HAdjustmentValue`.

---

**removeAdjustmentListener(HAdjustmentListener)**

```
public void removeAdjustmentListener(HAdjustmentListener l)
```

Removes the specified `HAdjustmentListener` so that it no longer receives `HAdjustmentEvent` from this object. If the specified listener is not registered, the method has no effect. If multiple references to a single listener have been registered it should be noted that this method will only remove one reference per call.

**Parameters:**

1 - the `HAdjustmentListener` to be removed from notification.

---

**setAdjustmentSound(HSound)**

```
public void setAdjustmentSound(HSound sound)
```

Associate a sound to be played when the value is modified. The sound is played irrespective of whether an `HAdjustmentEvent` is sent to one or more listeners.

**Parameters:**

`sound` - the sound to be played, when the value is modified. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

---

**setBlockIncrement(int)**

```
public void setBlockIncrement(int increment)
```



Set the block increment for this `HAdjustmentValue` .

**Parameters:**

`increment` - the amount by which the value of the `HAdjustmentValue` should change when an `ADJUST_PAGE_LESS` or `ADJUST_PAGE_MORE` event is received. Values of `increment` less than one shall be treated as a value of one.

---

**setUnitIncrement(int)**

```
public void setUnitIncrement(int increment)
```

Set the unit increment for this `HAdjustmentValue` .

**Parameters:**

`increment` - the amount by which the value of the `HAdjustmentValue` should change when an `ADJUST_LESS` or `ADJUST_MORE` event is received. Values of `increment` less than one shall be treated as a value of one.

# org.havi.ui

# HAnimateEffect

## Syntax

```
public interface HAnimateEffect
```

## All Known Implementing Classes:

```
HStaticAnimation, HFlatEffectMatte, HImageEffectMatte
```

## Description

The `HAnimateEffect` interface defines effect constants and controls for time-varying animations.

Implementations of `HAnimateEffect` should have the following default behaviors:

- By default the `HAnimateEffect` should be stopped. Hence, to start an `HAnimateEffect` the `start()` method must be explicitly invoked. This mechanism allows for animations that are programmatically controlled, e.g. via the `setPosition(int)` method.
- By default the position for rendering should be the first image in the sequence, i.e. 0.
- By default the play mode should be `PLAY_REPEATING`.
- By default the repeat count should be `REPEAT_INFINITE`.
- The default rendering should simply display the single image at the current position of the animation within the sequence.

## Fields

---

### PLAY\_ALTERNATING

```
public static final int PLAY_ALTERNATING
```

Indicates that the animation should be played in a repeating loop, alternating between the forward and reverse direction.

The images are rendered in the same order that they are present in the sequence (array):

```
0, 1, 2, 3, ... length-2, length-1
```

If the animation has not repeated sufficiently, then the rendering of the sequence is reversed - i.e. the images are rendered in the order

```
length-2, length-3, ... 1, 0
```

If the animation has not repeated sufficiently, then the rendering of the sequence is reversed (again) back to a forwards direction. I.e. the images are rendered in the order

```
1, 2, 3, ... length-2, length-1
```

Each rendering of the sequence of images forwards or backwards, should be considered as a single "repeat".

Note that when the sequence repeats, the last image (first image) is not rendered consecutively, i.e. twice.

---

### PLAY\_REPEATING

```
public static final int PLAY_REPEATING
```

Indicates that the animation should be played forwards (in a repeating loop).

The images are rendered in the same order that they are present in the sequence (array):

0, 1, 2, 3, ... length-1

If the animation has not repeated sufficiently, then the rendering of the sequence is restarted from the first image, i.e. the images will continue to be rendered in the order:

0, 1, 2, 3, ... length-1

Each rendering of the sequence of images 0 to (length-1), should be considered as a single "repeat".

---

## REPEAT\_INFINITE

```
public static final int REPEAT_INFINITE
```

This value, when passed to `setRepeatCount`, indicates that the animation shall repeat until the `stop()` method is invoked.

## Methods

---

### getDelay()

```
public int getDelay()
```

Gets the presentation delay for this `HAnimateEffect` .

**Returns:**

the presentation delay in units of 0.1 seconds.

---

### getPlayMode()

```
public int getPlayMode()
```

Gets the playing mode for this `HAnimateEffect` .

**Returns:**

the play mode for this `HAnimateEffect` .

---

### getPosition()

```
public int getPosition()
```

Get the current index into the content array which this `HAnimateEffect` is using to display content.

**Returns:**

the index of the content currently being displayed, in the range  $0 \leq \text{index} < \text{length}$

---

### getRepeatCount()

```
public int getRepeatCount()
```

Gets the number of times that this `HAnimateEffect` is to be played. Note that this method does *not* return the number of repeats that are remaining to be played.

**Returns:**

the total number of times that an `HAnimateEffect` is to be played. The returned value shall be greater than zero, or `REPEAT_INFINITE` .

---

**isAnimated()**

```
public boolean isAnimated()
```

This method indicates the animation (running) state of the `HAnimateEffect`.

**Returns:**

`true` if this `HAnimateEffect` is running, i.e. the `start` method has been invoked - `false` otherwise.

---

**setDelay(int)**

```
public void setDelay(int count)
```

Sets the delay between the presentation of successive pieces of content (frames).

After calling `setDelay(int)` on a currently playing `HAnimateEffect`, there is no guarantee that one or more frames will not be displayed using the previous delay until the new delay value takes effect.

**Parameters:**

`count` - the content presentation delay in units of 0.1 seconds duration. If `count` is less than one "unit", then it shall be treated as if it were a delay of one "unit", i.e. 0.1 seconds.

---

**setPlayMode(int)**

```
public void setPlayMode(int mode)
```

Sets the playing mode for this `HAnimateEffect`. If the animation is already running a call to `setPlayMode` will change the current value and affect the animation immediately. The position of the animation is unchanged.

**Parameters:**

`mode` - the play mode for this `HAnimateEffect`, which must be either `PLAY_ALTERNATING` or `PLAY_REPEATING`.

---

**setPosition(int)**

```
public void setPosition(int position)
```

Set this `HAnimateEffect` to display the content at the specified position. If the animation is already running a call to `setPosition` will change the current value and affect the animation immediately.

**Parameters:**

`position` - an index into the content array which specifies the next piece of content to be displayed. If `position` is less than 0, then the array element at index 0 is displayed, if `position` is greater than or equal to the length of the content array, then the array element at index `[length-1]` will be used.

---

**setRepeatCount(int)**

```
public void setRepeatCount(int count)
```

Sets the number of times that this `HAnimateEffect` should be played. If the animation is already running a call to `setRepeatCount` will change the current value and reset the current number of repeats to 0, affecting the animation immediately.

**Parameters:**

`count` - the number of times that an `HAnimateEffect` should be played. Valid values of the repeat count are one or more, and `REPEAT_INFINITE` .

---

**start()**

```
public void start()
```

This method starts this `HAnimateEffect` playing. If `start` is called when the animation is already running it resets the animation according to the current play mode, as returned by `getPlayMode()` .

---

**stop()**

```
public void stop()
```

This method indicates that the running `HAnimateEffect` should be stopped. After calling this method, there is no guarantee that one or more frames will not be displayed before the animation actually stops playing. If the animation is already stopped further calls to `stop` have no effect.

# org.havi.ui

# HAnimateLook

## Syntax

```
public class HAnimateLook implements HLook
```

```
java.lang.Object
|
+--org.havi.ui.HAnimateLook
```

## All Implemented Interfaces:

java.lang.Cloneable, HLook

## Description

The `HAnimateLook` class is used by any `HVisible` component to display animated graphical content. This look will be provided by the platform and the exact way in which it is rendered will be platform dependant.

The `HAnimateLook` class draws the content set on an `HVisible`. It uses the `getAnimateContent(int)` `getAnimateContent(int)`

- If the sequence is empty then the animation shall be treated as a completely transparent area of specific width and height within its enclosing Container --- simply for the purposes of layout management (if applicable).
- If there is only one image referenced in the sequence then that image is rendered statically in a similar manner to `HGraphicLook`.
- If there is more than one image referenced in the sequence then these are rendered in sequence, giving the effect of an animation.

If a referenced image is inaccessible, then it shall be skipped. If no images are accessible, then the animation shall be treated as a completely transparent area of specific width and height within its enclosing Container --- for the purposes of layout management (if applicable).

The `HAnimateLook` is not required to present consecutive images in the animation with the delay specified in its associated `HStaticAnimation`. For example, if the time taken to retrieve or render an image is longer than the delay, then then it shall be rendered as soon as possible.

Implementations of `HAnimateLook` should use the appropriate methods on `HVisible` to determine which scaling and alignment modes to use when rendering content. See the class description for `HLook` for more details. `HAnimateLook` may support scalable animated graphical content. As a minimum, all implementations must support the `RESIZE_NONE` scaling mode, and all alignment modes.

Note that the results of applying the `VALIGN_JUSTIFY` and `HALIGN_JUSTIFY` alignment modes for animated graphical content are defined to identical to `VALIGN_CENTER` and `HALIGN_CENTER` modes respectively, as justification is meaningless in this context.

This is the default look that is used by `HStaticAnimation` and its subclasses. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

See Also:

`HStaticAnimation`, `HVisible`, `HLook`

## Constructors

### HAnimateLook()

```
public HAnimateLook()
```

Creates a `HAnimateLook` object. See the class description for details of constructor parameters and default values.

## Methods

### getInsets(HVisible)

```
public java.awt.Insets getInsets(HVisible visible)
```

Determines the insets of this `HLook`, which indicate the size of the border. This area is reserved for the `HLook` to use for drawing borders around the associated `HVisible`.

**Specified By:**

`getInsets(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

the insets of this `HLook`.

### getMaximumSize(HVisible)

```
public java.awt.Dimension getMaximumSize(HVisible hvisible)
```

Gets the maximum size of the `HVisible` component when drawn with this `HLook`.

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook`) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If the `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) then the return value is the current size of the `HVisible` (as returned by `HVisible#getSize`).
2. If the `HLook` does not support scaling of content or no scaling is requested, and content is set then the

return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.

3. If there is no content set then a maximum size of `[ Short.MAX_VALUE, Short.MAX_VALUE ]` is returned as a `Dimension`.

**Specified By:**

`getMaximumSize(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating this `HLook` maximum size.

**See Also:**

`getMaximumSize()`

### **getMinimumSize(HVisible)**

```
public java.awt.Dimension getMinimumSize(HVisible hvisible)
```

Gets the minimum size of the `HVisible` component when drawn with this `HLook`.

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook`) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If the `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) and content is set then the return value is the size of the smallest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
2. If the `HLook` does not support scaling of content or no scaling is requested, *and* content is set then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If no content is available but a default preferred size has been set using `setDefaultSize(Dimension)` has been called to set then the return value is this value (as obtained with `getDefaultSize()`) plus any additional dimensions that the `HLook` requires for border decoration etc.
4. If there is no content or default size set then the return value is an implementation-specific minimum size plus any additional dimensions that the `HLook` requires for border decoration etc.

**Specified By:**

`getMinimumSize(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating this `HLook` minimum size.

**See Also:**

`getMinimumSize()`

### **getPreferredSize(HVisible)**

```
public java.awt.Dimension getPreferredSize(HVisible hvisible)
```

Gets the preferred size of the `HVisible` component when drawn with this `HLook`.



This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook`) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If a default preferred size has been set for this `HVisible` (using `setDefaultSize(Dimension)`) then the return value is this size (obtained with `getDefaultSize()`) plus any additional dimensions that the `HLook` requires for border decoration etc.
2. If this `HLook` does not support scaling of content or no scaling is requested, and content is present then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If this `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) and content is set then the return value is the current size of the `HVisible` as returned by `getSize()`.
4. If there is no content and no default size set then the return value is the current size of the `HVisible` as returned by `getSize()`.

**Specified By:**

`getPreferredSize(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating the preferred size of the `HVisible` when drawn with this `HLook`.

**See Also:**

`getPreferredSize()`, `setDefaultSize(Dimension)`

### **isOpaque(HVisible)**

```
public boolean isOpaque(HVisible visible)
```

Returns true if the entire painted area of the `HVisible` when using this look is fully opaque, i.e. the `showLook(Graphics, HVisible, int)` method guarantees that all pixels are painted in an opaque Color.

The default value is implementation specific and depends on the background painting mode of the given `HVisible`. The consequences of an invalid overridden value are implementation specific.

**Specified By:**

`isOpaque(HVisible)` in interface `HLook`

**Parameters:**

`visible` - the visible to test

**Returns:**

true if all the pixels with the `java.awt.Component#getBounds` method of an `HVisible` using this look are fully opaque, i.e. the `showLook(Graphics, HVisible, int)` method guarantees that all pixels are painted in an opaque Color, otherwise false.

### **showLook(Graphics, HVisible, int)**

```
public void showLook(java.awt.Graphics g, HVisible visible, int state)
```

The `showLook(Graphics, HVisible, int)` method is responsible for repainting the entire `HVisible` component, (including any content set on the component, and the component background), subject to the clipRect of the Graphics object passed to it.

The `showLook(Graphics, HVisible, int)` method should not modify the `clipRect` of the `Graphics` object that is passed to it.

For looks which draw content (e.g. `HTextLook`, `HGraphicLook` and `HAnimateLook`), if no content is associated with the component, the `showLook(Graphics, HVisible, int)` method paints the component with its current background `Color` according to the `setBackgroundMode(int)` method of `HVisible` and draws any (implementation-specific) borders. Note that by default the background mode is set so as to **not** paint a background. Furthermore on platforms which support transparent colors the background `Color` may be partially or completely transparent.

Any resources **explicitly** associated with an `HLook` should be loaded by the `HLook` during its creation, etc., or via its `setXXX()` methods. Note that the "standard" looks don't load content by default.

This method is called from the `paint(Graphics)` method of `HVisible` and must never be called from elsewhere. Components wishing to redraw themselves should call their `repaint` method in the usual way.

**Specified By:**

`showLook(Graphics, HVisible, int)` in interface `HLook`

**Parameters:**

`g` - the graphics context.

`visible` - the visible.

`state` - the state parameter indicates the state of the visible, allowing the look to render the appropriate content for that state. Note that some components (e.g. `HStaticRange`, `HRange`, `HRangeValue`) do not use state-based content).

**widgetChanged(HVisible, HChangeData[])**

```
public void widgetChanged(HVisible visible, HChangeData[] changes)
```

Called by the `HVisible` whenever its content, state, or any other data changes. See the class description of `HVisible` for more information about the `changes` parameter.

The implementation of this method should work out which graphical areas of the `HVisible` have changed and make any relevant calls to trigger the repainting of those areas.

A minimum implementation of this method could simply call

```
visible.repaint()
```

**Specified By:**

`widgetChanged(HVisible, HChangeData[])` in interface `HLook`

**Parameters:**

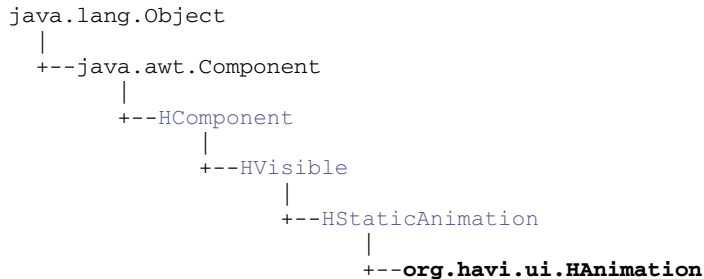
`visible` - the `HVisible` which has changed

`changes` - an array containing hint data and associated hint objects. If this argument is `null` a full repaint will be triggered.

# org.havi.ui HAnimation

## Syntax

public class HAnimation extends HStaticAnimation implements HNavigable



## All Implemented Interfaces:

HAnimateEffect, HMatteLayer, HNavigable, HNavigationInputPreferred, HNoInputPreferred, HState, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, TestOpacity

## Description

The `HAnimation` class is a user interface component used to display animated graphical content (as `HStaticAnimation`) which additionally enables a user to navigate to it, i.e. it can have the input focus. By default it uses the `HAnimateLook` to render itself. The `HAnimation` class supports animating images in exactly the same way as the `HStaticAnimation` class, with the following exception:

- If the `HAnimation` has different content set for the supported states a change in interaction state may cause an implementation-specific change to the animation position to deal with animation content of different length.

The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
height	height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

images	The array of images to be used as the content for every state of this component.	null	setAnimateContent (Image[ ], int)	getAnimateContent (int)
images-Normal	The array of images to be used as the content for the <code>NORMAL_STATE</code> state of this component.	null	setAnimateContent (Image[ ], int)	getAnimateContent (int)
imagesFocused	The array of images to be used as the content for the focused states of this component.	null	setAnimateContent (Image[ ], int)	getAnimateContent (int)
delay	The delay between the presentation of successive content in the animation, in units of 0.1 second	1 (i.e. 0.1 seconds)	setDelay (int)	getDelay ()
repeat-Count	The number of times that the animation is to be played.	REPEAT_INFINITE	setRepeatCount (int)	getRepeatCount ()
playMode	The playing mode for the animation.	PLAY_REPEATING	setPlayMode (int)	getPlayMode ()

#### Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte ( <code>HMatte</code> ).	none (i.e. <code>getMatte()</code> returns null)	<code>setMatte (HMatte)</code>	<code>getMatte ()</code>
The text layout manager responsible for text formatting.	An <code>HDefaultTextLayoutManager</code> object.	<code>setTextLayoutManager (HTextLayoutManager)</code>	<code>getTextLayoutManager ()</code>
The background painting mode	<code>NO_BACKGROUND_FILL</code>	<code>setBackgroundMode (int)</code>	<code>getBackgroundMode ()</code>
The default preferred size	not set (i.e. null) unless specified by width and height parameters	<code>setDefaultSize (Dimension)</code>	<code>getDefaultSize ()</code>
The horizontal content alignment	<code>HALIGN_CENTER</code>	<code>setHorizontalAlignment (int)</code>	<code>getHorizontalAlignment ()</code>
The vertical content alignment	<code>VALIGN_CENTER</code>	<code>setVerticalAlignment (int)</code>	<code>getVerticalAlignment ()</code>
The content scaling mode	<code>RESIZE_NONE</code>	<code>setResizeMode (int)</code>	<code>getResizeMode ()</code>
The initial piece of content to be presented, i.e. its position in the content array.	0	<code>setPosition (int)</code>	<code>getPosition ()</code>

By default the animation should be stopped. Hence, to start the animation its start method must be explicitly invoked. This mechanism allows for animations that are programmatically controlled, e.g. via the setPosition method.	"stopped"	<code>start() / stop()</code>	<code>isAnimated()</code>
The default "look" for this class.	A platform specific <code>HAnimateLook</code>	<code>setDefaultLook( HAnimateLook)</code>	<code>getDefaultLook( )</code>
The "look" for this object.	The <code>HAnimateLook</code> returned from <code>HAnimation.getDefaultLook</code> when this object was created.	<code>setLook(HLook)</code>	<code>getLook()</code>
The gain focus sound.	null	<code>setGainFocusSou nd(HSound)</code>	<code>getGainFocusSou nd()</code>
The lose focus sound.	null	<code>setLoseFocusSou nd(HSound)</code>	<code>getLoseFocusSou nd()</code>

**See Also:**

`HStaticAnimation`, `HNavigable`

## Constructors

---

**HAnimation()**

```
public HAnimation()
```

Creates an `HAnimation` object. See the class description for details of constructor parameters and default values.

---

**HAnimation(Image[], Image[], int, int, int)**

```
public HAnimation(java.awt.Image[] imagesNormal, java.awt.Image[] imagesFocused, int delay,  
int playMode, int repeatCount)
```

Creates an `HAnimation` object. See the class description for details of constructor parameters and default values.

---

**HAnimation(Image[], Image[], int, int, int, int, int, int, int)**

```
public HAnimation(java.awt.Image[] imagesNormal, java.awt.Image[] imagesFocused, int delay,  
int playMode, int repeatCount, int x, int y, int width, int height)
```

Creates an `HAnimation` object. See the class description for details of constructor parameters and default values.

---

**HAnimation(Image[], int, int, int)**

```
public HAnimation(java.awt.Image[] images, int delay, int playMode, int repeatCount)
```

Creates an HAnimation object. See the class description for details of constructor parameters and default values.

---

**HAnimation(Image[], int, int, int, int, int, int)**

```
public HAnimation(java.awt.Image[] images, int delay, int playMode, int repeatCount, int x,
                  int y, int width, int height)
```

Creates an HAnimation object. See the class description for details of constructor parameters and default values.

## Methods

---

**getDefaultLook()**

```
public static HAnimateLook getDefaultLook()
```

Returns the currently set default look for HAnimation components.

**Returns:**

The [HLook](#) that is used by default when creating a new HAnimation component.

---

**getGainFocusSound()**

```
public HSound getGainFocusSound()
```

Get the sound associated with the gain focus event.

**Specified By:**

[getGainFocusSound\(\)](#) in interface [HNavigable](#)

**Returns:**

The sound played when the component gains focus. If no sound is associated with gaining focus, then null shall be returned.

---

**getLoseFocusSound()**

```
public HSound getLoseFocusSound()
```

Get the sound associated with the lost focus event.

**Specified By:**

[getLoseFocusSound\(\)](#) in interface [HNavigable](#)

**Returns:**

The sound played when the component loses focus. If no sound is associated with losing focus, then null shall be returned.

---

**getMove(int)**

```
public HNavigable getMove(int keyCode)
```

Provides the [HNavigable](#) object that is navigated to when a particular key is pressed.

**Specified By:**

`getMove(int)` in interface `HNavigable`

**Parameters:**

`keyCode` - The key code of the pressed key.

**Returns:**

Returns the `HNavigable` object, or if no `HNavigable` is associated with the `keyCode` then returns `null`.

**getNavigationKeys()**

`public int[] getNavigationKeys()`

Retrieve the set of key codes which this component maps to navigation targets.

**Specified By:**

`getNavigationKeys()` in interface `HNavigationInputPreferred`

**Returns:**

an array of key codes, or `null` if no navigation targets are set on this component.

**isSelected()**

`public boolean isSelected()`

Indicates if this component has focus.

**Specified By:**

`isSelected()` in interface `HNavigable`

**Returns:**

`true` if the component has focus, otherwise returns `false`.

**processHFocusEvent(HFocusEvent)**

`public void processHFocusEvent(HFocusEvent evt)`

Process an `HFocusEvent` sent to this `HAnimation`.

**Specified By:**

`processHFocusEvent(HFocusEvent)` in interface `HNavigationInputPreferred`

**Parameters:**

`evt` - the `HFocusEvent` to process.

**setDefaultLook(HAnimateLook)**

`public static void setDefaultLook(HAnimateLook hlook)`

Sets the default `HLook` for further `HAnimation` Components.

**Parameters:**

`hlook` - The `HLook` that will be used by default when creating a new `HAnimation` component. Note that this parameter may be `null`, in which case newly created components shall not draw themselves until a non-`null` look is set using the `setLook(HLook)` method.

**setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)**

`public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left, HNavigable right)`

Set the focus control for an `HNavigable` component. Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is a convenience function for application programmers where a standard up, down, left and right focus traversal between components is required.

Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

Note that this API does not prevent the creation of "isolated" `HNavigable` components --- authors should endeavor to avoid confusing the user.

**Specified By:**

`setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` in interface `HNavigable`

**Parameters:**

`up` - The `HNavigable` component to move to, when the user generates a `VK_UP` KeyEvent. If there is no `HNavigable` component to move "up" to, then null should be specified.

`down` - The `HNavigable` component to move to, when the user generates a `VK_DOWN` KeyEvent. If there is no `HNavigable` component to move "down" to, then null should be specified.

`left` - The `HNavigable` component to move to, when the user generates a `VK_LEFT` KeyEvent. If there is no `HNavigable` component to move "left" to, then null should be specified.

`right` - The `HNavigable` component to move to, when the user generates a `VK_RIGHT` KeyEvent. If there is no `HNavigable` component to move "right" to, then null should be specified.

### **setGainFocusSound(HSound)**

```
public void setGainFocusSound(HSound sound)
```

Associate a sound with gaining focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_GAINED`. This sound will start to be played when an object implementing this interface gains focus. It is not guaranteed to be played to completion. If the object implementing this interface loses focus before the audio completes playing, the audio will be truncated. Applications wishing to ensure the audio is always played to completion must implement special logic to slow down the focus transitions.

By default, an `HNavigable` object does not have any gain focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

**Specified By:**

`setGainFocusSound(HSound)` in interface `HNavigable`

**Parameters:**

`sound` - the sound to be played, when the component gains focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

### **setLoseFocusSound(HSound)**

```
public void setLoseFocusSound(HSound sound)
```

Associate a sound with losing focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_LOST`. This sound will start to be played when an object implementing this interface loses focus. It is not guaranteed to be played to completion. It is implementation dependent whether and when this sound will be truncated by any gain focus sound played by the next object to gain focus.



By default, an `HNavigable` object does not have any lose focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

**Specified By:**

`setLoseFocusSound(HSound)` in interface `HNavigable`

**Parameters:**

`sound` - the sound to be played, when the component loses focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

---

**setMove(int, HNavigable)**

```
public void setMove(int keyCode, HNavigable target)
```

Defines the navigation path from the current `HNavigable` to another `HNavigable` when a particular key is pressed.

Note that `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

**Specified By:**

`setMove(int, HNavigable)` in interface `HNavigable`

**Parameters:**

`keyCode` - The key code of the pressed key. Any numerical keycode is allowed, but the platform may not be able to generate all keycodes. Application authors should only use keys for which `HRcCapabilities.isSupported()` or `HKeyCapabilities.isSupported()` returns true.

`target` - The target `HNavigable` object that should be navigated to. If a target is to be removed from a particular navigation path, then `null` should be specified.

# org.havi.ui HBackgroundConfigTemplate

## Syntax

```
public class HBackgroundConfigTemplate extends HScreenConfigTemplate
```

```
java.lang.Object
|
+--HScreenConfigTemplate
|
+--org.havi.ui.HBackgroundConfigTemplate
```

## Description

The `HBackgroundConfigTemplate` class is used to obtain a valid `HBackgroundConfiguration`. An application instantiates one of these objects and then sets all non-default attributes as desired. The `getBestConfiguration(HBackgroundConfigTemplate)` method found in the `HBackgroundDevice` class is then called with this `HBackgroundConfigTemplate`. A valid `HBackgroundConfiguration` is returned that meets or exceeds what was requested in the `HBackgroundConfigTemplate`.

This class may be subclassed to support additional properties of background configurations which may be requested by applications.

The `ZERO_VIDEO_IMPACT` property may be used in instances of this class to discover whether displaying background stills will have any impact on already running video. Implementations supporting the `STILL_IMAGE` preference shall return an `HStillImageBackgroundConfiguration` when requested except as described below.

- If displaying an `STILL_IMAGE` interrupts video transiently while the image is decoded then a configuration shall not be returned if the `ZERO_VIDEO_IMPACT` property is present with the priority `REQUIRED`.
- If displaying an `STILL_IMAGE` interrupts video while the image is decoded and for the entire period while the image is displayed then a configuration shall not be returned if the `ZERO_VIDEO_IMPACT` property is present with either the priorities `REQUIRED` or `PREFERRED`.

The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
-----------	-------------	---------------	------------	------------

None.

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

None.

## See Also:

`HScreenConfigTemplate`, `HGraphicsConfigTemplate`, `HVideoConfigTemplate`

## Fields

---

## CHANGEABLE\_SINGLE\_COLOR

```
public static final int CHANGEABLE_SINGLE_COLOR
```

A value for use in the preference field of the `setPreference(int, int)` and `getPreferencePriority(int)` methods in the `HBackgroundConfigTemplate` that indicates that a single color background is requested where that single color can be changed by applications.

---

## STILL\_IMAGE

```
public static final int STILL_IMAGE
```

A value for use in the preference field of the `setPreference(int, int)` and `getPreferencePriority(int)` methods in the `HBackgroundConfigTemplate` that indicates that a background which can support still images is requested. Where backgrounds supporting this feature are returned, they are returned as objects of the `HStillImageBackgroundConfiguration` class.

## Constructors

---

### HBackgroundConfigTemplate()

```
public HBackgroundConfigTemplate()
```

Creates an `HBackgroundConfigTemplate` object. See the class description for details of constructor parameters and default values.

## Methods

---

### getPreferencePriority(int)

```
public int getPreferencePriority(int preference)
```

Return the priority for the specified preference.

By default the preferences in a template returned from the system will have an `DONT_CARE` priority unless specified otherwise. Any configuration always satisfies these attributes.

**Overrides:**

`getPreferencePriority(int)` in class `HScreenConfigTemplate`

**Parameters:**

`preference` - the preference to be indicated. Valid values for an `HBackgroundConfigTemplate` are: `ZERO_BACKGROUND_IMPACT`, `ZERO_GRAPHICS_IMPACT`, `ZERO_VIDEO_IMPACT`, `INTERLACED_DISPLAY`, `FLICKER_FILTERING`, `VIDEO_GRAPHICS_PIXEL_ALIGNED.PIXEL_ASPECT_RATIO`, `PIXEL_RESOLUTION`, `SCREEN_RECTANGLE`, `CHANGEABLE_SINGLE_COLOR` and `STILL_IMAGE`.

Subclasses may add further valid values. An `IllegalArgumentException` shall be thrown if the preference is not a valid value for this instance of `HBackgroundConfigTemplate`

**Returns:**

the priority for the specified preference.

---

**isConfigSupported(HBackgroundConfiguration)**

```
public boolean isConfigSupported(HBackgroundConfiguration hbc)
```

Returns a boolean indicating whether or not the specified `HBackgroundConfiguration` can be used to create a background plane that supports the features set in this template.

**Parameters:**

`hbc` - - the `HBackgroundConfiguration` object to test against this template.

**Returns:**

true if this `HBackgroundConfiguration` object can be used to create a background plane that supports the features set in this template, false otherwise.

---

**setPreference(int, int)**

```
public void setPreference(int preference, int priority)
```

Set the indicated preference to have the specified priority. If the preference has been previously set, then the previous priority for the preference shall be overwritten.

Attributes that are not filled in in a template (through `setPreference(int, int)`), shall have the priority `DONT_CARE`. Any configuration always satisfies these attributes.

**Overrides:**

`setPreference(int, int)` in class `HScreenConfigTemplate`

**Parameters:**

`preference` - the preference to be indicated. Valid values for an `HBackgroundConfigTemplate` are: `ZERO_BACKGROUND_IMPACT`, `ZERO_GRAPHICS_IMPACT`, `ZERO_VIDEO_IMPACT`, `INTERLACED_DISPLAY`, `FLICKER_FILTERING`, `VIDEO_GRAPHICS_PIXEL_ALIGNED`, `CHANGEABLE_SINGLE_COLOR` and `STILL_IMAGE`.

Subclasses may add further valid values. An `IllegalArgumentException` shall be thrown if the preference is not a valid value for this instance of `HBackgroundConfigTemplate`

`priority` - the priority of the preference. Valid values are: `REQUIRED`, `PREFERRED`, `DONT_CARE`, `PREFERRED_NOT` and `REQUIRED_NOT`.

If `priority` is not a valid priority as defined here a `java.lang.IllegalArgumentException` will be thrown.

# org.havi.ui HBackgroundConfiguration

## Syntax

```
public class HBackgroundConfiguration extends HScreenConfiguration
```

```
java.lang.Object
|
+--HScreenConfiguration
|
+--org.havi.ui.HBackgroundConfiguration
```

## Direct Known Subclasses:

```
HStillImageBackgroundConfiguration
```

## Description

The `HBackgroundConfiguration` class describes the characteristics (settings) of an `HBackgroundDevice`. There can be many `HBackgroundConfiguration` objects associated with a single `HBackgroundDevice`.

The basic background configuration supports backgrounds of a single color. More sophisticated backgrounds can be supported by defining new classes inheriting from this class. Where a device has a single non-changeable background color, this class will provide applications the ability to read that color however all attempts to reserve control of the background will fail. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
-----------	-------------	---------------	------------	------------

None.

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

None.

## See Also:

[HBackgroundDevice](#)

## Constructors

### HBackgroundConfiguration()

```
protected HBackgroundConfiguration()
```

It is not intended that applications should directly construct `HBackgroundConfiguration` objects.

Creates an `HBackgroundConfiguration` object. See the class description for details of constructor parameters and default values.

## Methods

---

### `getColor()`

```
public java.awt.Color getColor()
```

Obtain the current color of this background. This method may be called without ownership of the resource. The value returned is not guaranteed to be the value set in the last call to `setColor(Color)` since platforms may offer a reduced color space for backgrounds and the actual value used will be returned.

**Returns:**

the current `Color`

---

### `getConfigTemplate()`

```
public HBackgroundConfigTemplate getConfigTemplate()
```

Returns an `HBackgroundConfigTemplate` object that describes and uniquely identifies this `HBackgroundConfiguration`. Hence, the following sequence should return the original `HBackgroundConfiguration`

```
HBackgroundDevice.getBestMatch(HBackgroundConfiguration.getConfigTemplate())
```

Features that are implemented in the `HBackgroundConfiguration` will return `REQUIRED` priority. Features that are not implemented in the `HBackgroundConfiguration` will return `REQUIRED_NOT` priority.

**Returns:**

an `HBackgroundConfigTemplate` object which both describes and uniquely identifies this `HBackgroundConfiguration`.

---

### `getDevice()`

```
public HBackgroundDevice getDevice()
```

Returns the `HBackgroundDevice` associated with this `HBackgroundConfiguration`.

**Returns:**

the `HBackgroundDevice` object that is associated with this `HBackgroundConfiguration`.

---

### `setColor(Color)`

```
public void setColor(java.awt.Color color)
```

Set the current color of this background. On platforms where there is a sub-class of `java.awt.Color` supporting transparency of any kind, passing an object representing a non-opaque color is illegal. Platforms with a limited color resolution for backgrounds may approximate this value to the nearest available. The `getColor()` method will return the actual value used.

**Parameters:**

`color` - the color to be used for the background

**Throws:**

`HPermissionDeniedException` - if this `HBackgroundDevice` does not have the right to control the background

`HConfigurationException` - if the color specified is illegal for this platform.

# org.havi.ui HBackgroundDevice

## Syntax

```
public class HBackgroundDevice extends HScreenDevice
```

```
java.lang.Object
|
+--HScreenDevice
|
+--org.havi.ui.HBackgroundDevice
```

## All Implemented Interfaces:

org.davic.resources.ResourceProxy, org.davic.resources.ResourceServer

## Description

This class represents the ultimate background of a screen. The background is the very back of the video / graphics composition stack. It can potentially cover the entire area of a screen. Where a device supports multiple applications on screen at the same time (or even a window manager), the background is not constrained by any particular application or window. The right to control the background of a screen is a scarce resource and managed as such. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Constructors

### HBackgroundDevice()

```
protected HBackgroundDevice()
```

It is not intended that applications should directly construct `HBackgroundDevice` objects.

Creates an `HBackgroundDevice` object. See the class description for details of constructor parameters and default values.

## Methods

---

**getBestConfiguration(HBackgroundConfigTemplate)**

```
public HBackgroundConfiguration getBestConfiguration(HBackgroundConfigTemplate hbc)
```

Returns the "best" configuration possible that passes the criteria defined in this `HBackgroundConfigTemplate` or null.

Equally best in this sense means that the configurations satisfy an equal number of preferences with priorities `PREFERRED` and `PREFERRED_NOT` and all preferences with priorities `REQUIRED` and `REQUIRED_NOT`.

Configurations are chosen according to the following algorithm, based on the priority as supplied to `setPreference(int, int)`. Configurations must:

1. satisfy ALL the preferences whose priority was `REQUIRED`
2. satisfy NONE of the preferences whose priority was `REQUIRED_NOT`
3. satisfy as many as possible of the preferences whose priority was `PREFERRED`.
4. satisfy as few as possible of the preferences whose priority was `PREFERRED_NOT`.

Preferences whose priority was `DONT_CARE` are ignored.

This method returns null if no configuration exists that satisfies all `REQUIRED` and `REQUIRED_NOT` priorities.

**Parameters:**

`hbc` - - an `HBackgroundConfigTemplate` object used to obtain a valid `HBackgroundConfiguration`. If this parameter is null the default configuration for the platform shall be returned.

**Returns:**

an `HBackgroundConfiguration` object that passes the criteria defined in the specified `HGraphicsConfigTemplate` or null if no `HBackgroundConfiguration` passes the criteria.

---

**getBestConfiguration(HBackgroundConfigTemplate[])**

```
public HBackgroundConfiguration getBestConfiguration(HBackgroundConfigTemplate[] hbcta)
```

Returns the "best" configuration possible that passes the criteria defined in one of the `HBackgroundConfigTemplate` objects within the specified array or null. The `HBackgroundTemplate` objects should be considered for matching in priority order from 0 to `(hbcta.length - 1)`.

Equally best in this sense means that the configurations satisfy an equal number of preferences with priorities `PREFERRED` and `PREFERRED_NOT` and all preferences with priorities `REQUIRED` and `REQUIRED_NOT`.

Configurations are chosen according to the following algorithm, based on the priority as supplied to `setPreference(int, int)`. Configurations must:

1. satisfy ALL the preferences whose priority was `REQUIRED`
2. satisfy NONE of the preferences whose priority was `REQUIRED_NOT`
3. satisfy as many as possible of the preferences whose priority was `PREFERRED`.
4. satisfy as few as possible of the preferences whose priority was `PREFERRED_NOT`.

Preferences whose priority was `DONT_CARE` are ignored.

This method returns null if no configuration exists that satisfies all `REQUIRED` and `REQUIRED_NOT` priorities.



**Parameters:**

hbcta - the `HBackgroundConfigTemplate` array used to obtain a valid `HBackgroundConfiguration`.

**Returns:**

an `HBackgroundConfiguration` that passes the criteria defined in one of the `HBackgroundConfigTemplate` objects within the specified array

**getConfigurations()**

```
public HBackgroundConfiguration[] getConfigurations()
```

Returns all of the `HBackgroundConfiguration` objects associated with this `HBackgroundDevice`.

**Returns:**

an array of `HBackgroundConfiguration` objects

**See Also:**

`HBackgroundConfiguration`

**getCurrentConfiguration()**

```
public HBackgroundConfiguration getCurrentConfiguration()
```

Returns the current `HBackgroundConfiguration` for this `HBackgroundDevice`.

**Returns:**

the current `HBackgroundConfiguration` for this `HBackgroundDevice`.

**See Also:**

`HBackgroundConfiguration`

**getDefaultConfiguration()**

```
public HBackgroundConfiguration getDefaultConfiguration()
```

Returns the default `HBackgroundConfiguration` associated with this `HBackgroundDevice`. This (single) default configuration should correspond to some well-behaved settings for the device, such as, a minimal configuration or factory preset settings.

**Returns:**

the default `HBackgroundConfiguration` of this `HBackgroundDevice`

**See Also:**

`HBackgroundConfiguration`

**setBackgroundConfiguration(HBackgroundConfiguration)**

```
public boolean setBackgroundConfiguration(HBackgroundConfiguration hbc)
```

Set the background configuration for the device.

An application is only allowed to call this method after it reserved the device explicitly and subject to the security policy of the platform. Subject to this, the following rules determine whether this method can succeed.

- If the application selects a configuration that is not conflicting with the configurations of all the other devices on the same `HScreen` then that configuration is selected.
- If an application tries to select a configuration that conflicts with a configuration of another device on this `HScreen` which this application cannot control due to platform security policy then this method shall fail with a `SecurityException`.

- If an application tries to select a configuration that conflicts with a configuration of another device on this `HScreen` which this application cannot control due to another application owning the right to control that device and the platform not giving that right to this application then this method shall fail with an `HPermissionDeniedException`.
- If an application tries to select a configuration that conflicts with a configuration of another device which this application can control due to either this or no application having reserved the device, then the configuration of the other device is automatically changed. There is no change in the resource ownership of the other device.

Any modifications made to the configurations of other devices shall be reflected by the API for the devices concerned. The `getCurrentConfiguration()` method for those devices shall return the new configuration.

On successful change to the specified configuration, the device shall fire one or more `HScreenConfigurationEvent` for all listeners that are currently registered (if the criteria for receiving such an event has been satisfied). If the new configuration differs by more than one characteristic from the previous configuration then the device may fire one or more `HScreenConfigurationEvent`. This behavior is implementation specific.

If an attempt is made to set the identical configuration to that which is current, then no such event(s) shall be fired.

**Parameters:**

`hbc` - the `HBackgroundConfiguration` to which this device should be set.

**Returns:**

A boolean indicating whether the configuration could be applied successfully. If the configuration could not be applied successfully, the configuration after this method may not match the configuration of the device prior to this method being called --- applications should take steps to determine whether a partial change of settings has been made.

**Throws:**

`SecurityException` - if the application does not have sufficient rights to set the configuration for this device.

`HPermissionDeniedException` - if the application does not currently have the right to set the configuration for this device.

`HConfigurationException` - if the specified configuration is not valid for this device, or if it conflicts with other devices whose configuration(s) cannot be changed by this application.

# org.havi.ui HBackgroundImage

## Syntax

```
public class HBackgroundImage
```

```
java.lang.Object
|
+--org.havi.ui.HBackgroundImage
```

## Description

This class represents a background image. Images of this class can be used as full screen backgrounds outside the java.awt framework. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Constructors

### HBackgroundImage(byte[])

```
public HBackgroundImage(byte[] pixels)
```

Create an `HBackgroundImage` object from an array of bytes encoded in the same encoding format as when reading this type of image data from a file.

If this constructor succeeds then the object will automatically be in the loaded state and calling the `load(HBackgroundImageListener)` method shall immediately generate an `HBackgroundImageEvent` reporting success.

If the byte array does not contain a valid image then this constructor shall throw a `java.lang.IllegalArgumentException`.

Calling the `flush()` method on an object built with this constructor shall have no effect.

#### Parameters:

`pixels` - the data for the `HBackgroundImage` object encoded in the specified format for image files of this type.

### HBackgroundImage(String)

```
public HBackgroundImage(java.lang.String filename)
```

Create an `HBackgroundImage` object. Loading of the data for the object is not required at this time.

**Parameters:**

`filename` - the name of the file to use as the source of data in a platform-specific URL format.

---

### **HBackgroundImage(URL)**

```
public HBackgroundImage(java.net.URL contents)
```

Create an `HBackgroundImage` object. Loading of the data for the object is not required at this time.

**Parameters:**

`contents` - a URL referring to the data to load.

## **Methods**

---

### **flush()**

```
public void flush()
```

Flush all the resources used by this image. This includes any pixel data being cached as well as all underlying system resources used to store data or pixels for the image. After calling this method the image is in a state similar to when it was first created without any load method having been called. When this method is called, the image shall not be in use by an application. Resources related to any `HBackgroundDevice` are not released.

---

### **getHeight()**

```
public int getHeight()
```

Determines the height of the image. This is returned in pixels as defined by the format of the image concerned. If this information is not known when this method is called then -1 is returned.

The image must have been successfully loaded to completion before this information is guaranteed to be available. It is implementation specific whether this information is available before the image is successfully loaded to completion. An image whose loading failed for any reason shall be considered as having this information unavailable.

**Returns:**

the height of the image

---

### **getWidth()**

```
public int getWidth()
```

Determines the width of the image. This is returned in pixels as defined by the format of the image concerned. If this information is not known when this method is called then -1 is returned.

The image must have been successfully loaded to completion before this information is guaranteed to be available. It is implementation specific whether this information is available before the image is successfully loaded to completion. An image whose loading failed for any reason shall be considered as having this information unavailable.

**Returns:**

the width of the image

---

**load(HBackgroundImageListener)**

```
public void load(HBackgroundImageListener l)
```

Load the data for this object. This method is asynchronous. The completion of data loading is reported through the listener provided.

**Parameters:**

l - the listener to call when loading of data is completed.

# org.havi.ui HChangeData

## Syntax

```
public class HChangeData

java.lang.Object
|
+--org.havi.ui.HChangeData
```

## Description

A class used as argument for `widgetChanged(HVisible, HChangeData[])`. The hint constants are defined on `HVisible`.

## Fields

---

### data

```
public java.lang.Object data
```

The data object for this `HChangeData`. The types of this object for the different hints are defined on `HVisible`.

---

### hint

```
public int hint
```

The hint for this `HChangeData`. The hint constants are defined on `HVisible`.

## Constructors

---

### HChangeData(int, Object)

```
public HChangeData(int hint, java.lang.Object data)
```

Creates an `HChangeData` object.

#### Parameters:

`hint` - the hint constant for this `HChangeData`.

`data` - the data object for this `HChangeData`.

# org.havi.ui HComponent

## Syntax

public abstract class HComponent extends java.awt.Component implements HMatteLayer, TestOpacity

```
java.lang.Object
|
+--java.awt.Component
|
+--org.havi.ui.HComponent
```

## Direct Known Subclasses:

HVideoComponent, HVisible

## All Implemented Interfaces:

HMatteLayer, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, TestOpacity

## Description

The `HComponent` class extends the `java.awt.Component` class by implementing the `HMatteLayer` interface. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
height	height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte (HMatte).	none (i.e. getMatte() returns null)	setMatte(HMatte)	getMatte()

## Constructors

---

### HComponent()

```
public HComponent()
```

Creates an HComponent object. See the class description for details of constructor parameters and default values.

---

### HComponent(int, int, int, int)

```
public HComponent(int x, int y, int width, int height)
```

Creates an HComponent object. See the class description for details of constructor parameters and default values.

## Methods

---

### getMatte()

```
public HMatte getMatte()
```

Get any HMatte currently associated with this component.

**Specified By:**

`getMatte()` in interface HMatteLayer

**Returns:**

the HMatte currently associated with this component or null if there is no associated matte.

---

### isDoubleBuffered()

```
public boolean isDoubleBuffered()
```

Returns true if all the drawing done during the update and paint methods for this specific HComponent object is automatically double buffered.

**Overrides:**

`java.awt.Component.isDoubleBuffered()` in class `java.awt.Component`

**Returns:**

true if all the drawing done during the update and paint methods for this specific HComponent object is automatically double buffered, or false if drawing is not double buffered. The default value for the double buffering setting is platform-specific.

---

### isOpaque()

```
public boolean isOpaque()
```

Returns true if the entire HComponent area, as given by the `java.awt.Component#getBounds` method, is fully opaque, i.e. its paint method (or surrogate methods) guarantee that all pixels are painted in an opaque Color.

By default, the return value is `false`. The return value should be overridden by subclasses that can guarantee full opacity. The consequences of an invalid overridden value are implementation specific.

**Specified By:**



`isOpaque()` in interface `TestOpacity`

**Returns:**

`true` if all the pixels within the area given by the `java.awt.Component#getBounds` method are fully opaque, i.e. its paint method (or surrogate methods) guarantee that all pixels are painted in an opaque `Color`, otherwise `false`.

---

**processEvent(AWTEvent)**

```
protected void processEvent(java.awt.AWTEvent evt)
```

This method may be overridden to facilitate the generation of events in the `org.havi.ui.event` package from `java.awt.AWTEvents`. Events used in this way (e.g. `java.awt.event.KeyEvent`s) are NOT consumed and should be propagated to the superclass.

Subclasses of `HComponent` must always call the superclass if they override this function.

**Overrides:**

`java.awt.Component.processEvent(java.awt.AWTEvent)` in class `java.awt.Component`

**Parameters:**

`evt` - the `java.awt.AWTEvent` to handle.

---

**setMatte(HMatte)**

```
public void setMatte(HMatte m)
```

Applies an `HMatte` to this component, for matte compositing. Any existing animated matte must be stopped before this method is called or an `HMatteException` will be thrown.

**Specified By:**

`setMatte(HMatte)` in interface `HMatteLayer`

**Parameters:**

`m` - The `HMatte` to be applied to this component -- note that only one matte may be associated with the component, thus any previous matte will be replaced. If `m` is null, then any matte associated with the component is removed and further calls to `getMatte()` shall return null. The component shall behave as if it had a fully opaque `HFlatMatte` associated with it (i.e. an `HFlatMatte` with the default value of 1.0.)

**Throws:**

`HMatteException` - if the `HMatte` cannot be associated with the component. This can occur:

- if the specific matte type is not supported
- if the platform does not support any matte type
- if the component is associated with an already running `HFlatEffectMatte` or `HImageEffectMatte`. The exception is thrown even if `m` is null.

**See Also:**

`HMatte`

# org.havi.ui

# HComponentOrdering

## Syntax

```
public interface HComponentOrdering
```

## All Known Implementing Classes:

`HContainer`, `HScene`

## Description

This interface is implemented for all HAVi component containers that support the manipulation of the z-ordering of their children.

## See Also:

`HScene`, `HContainer`

## Methods

---

### addAfter(Component, Component)

```
public java.awt.Component addAfter(java.awt.Component component, java.awt.Component front)
```

Adds a `java.awt.Component` to this `HComponentOrdering` directly behind a previously added `java.awt.Component`.

If `component` has already been added to this container, then `addAfter` moves `component` behind `front`. If `front` and `component` are the same component which was already added to this container, `addAfter` does not change the ordering of the components and returns `component`.

This method affects the Z-order of the `java.awt.Component` children within the `HComponentOrdering`, and may also implicitly change the numeric ordering of those children.

#### Parameters:

`component` - is the `java.awt.Component` to be added to the `HComponentOrdering`

`front` - is the `java.awt.Component`, which `component` will be placed behind, i.e. `front` will be directly in front of the added `java.awt.Component`

#### Returns:

If the `java.awt.Component` is successfully added, then it will be returned from this call. If the `java.awt.Component` is not successfully added, e.g. `front` is not a `java.awt.Component` currently added to the `HComponentOrdering`, then `null` will be returned.

This method must be implemented in a thread safe manner.

---

### addBefore(Component, Component)

```
public java.awt.Component addBefore(java.awt.Component component,  
    java.awt.Component behind)
```

Adds a `java.awt.Component` to this `HComponentOrdering` directly in front of a previously added `java.awt.Component`.

If `component` has already been added to this container, then `addBefore` moves `component` in front of `behind`. If `behind` and `component` are the same component which was already added to this container, `addBefore` does not change the ordering of the components and returns `component`.

This method affects the Z-order of the `java.awt.Component` children within the `HComponentOrdering`, and may also implicitly change the numeric ordering of those children.

**Parameters:**

`component` - is the `java.awt.Component` to be added to the `HComponentOrdering`

`behind` - is the `java.awt.Component`, which `component` will be placed in front of, i.e. `behind` will be directly behind the added `java.awt.Component`

**Returns:**

If the `java.awt.Component` is successfully added, then it will be returned from this call. If the `java.awt.Component` is not successfully added, e.g. `behind` is not a `java.awt.Component` currently added to the `HComponentOrdering`, then `null` will be returned.

This method must be implemented in a thread safe manner.

### **pop(Component)**

```
public boolean pop(java.awt.Component component)
```

Moves the specified `java.awt.Component` one component nearer in the Z-order, i.e. swapping it with the `java.awt.Component` that was directly in front of it.

If `component` is already at the front of the Z-order, the order is unchanged and `pop` returns `true`.

**Parameters:**

`component` - The `java.awt.Component` to be moved.

**Returns:**

returns `true` on success, `false` on failure, for example if the `java.awt.Component` has yet to be added to the `HComponentOrdering`.

### **popInFrontOf(Component, Component)**

```
public boolean popInFrontOf(java.awt.Component move, java.awt.Component behind)
```

Puts the specified `java.awt.Component` in front of another `java.awt.Component` in the Z-order of this `HComponentOrdering`.

If `move` and `behind` are the same component which has been added to the container `popInFront` does not change the Z-order and returns `true`.

**Parameters:**

`move` - The `java.awt.Component` to be moved directly in front of the "behind" Component in the Z-order of this `HComponentOrdering`.

`behind` - The `java.awt.Component` which the "move" Component should be placed directly in front of.

**Returns:**

returns `true` on success, `false` on failure, for example when either `java.awt.Component` has yet to be added to the `HComponentOrdering`. If this method fails, the Z-order is unchanged.

### **popToFront(Component)**

```
public boolean popToFront(java.awt.Component component)
```

Brings the specified `java.awt.Component` to the "front" of the Z-order in this `HComponentOrdering`.

If component is already at the front of the Z-order, the order is unchanged and `popToFront` returns `true`.

**Parameters:**

`component` - The `java.awt.Component` to bring to the "front" of the Z-order of this `HComponentOrdering`.

**Returns:**

returns `true` on success, `false` on failure, for example when the `java.awt.Component` has yet to be added to the `HComponentOrdering`. If this method fails, the Z-order is unchanged.

---

### **push(Component)**

```
public boolean push(java.awt.Component component)
```

Moves the specified `java.awt.Component` one component further away in the Z-order, i.e. swapping it with the `java.awt.Component` that was directly behind it.

If component is already at the back of the Z-order, the order is unchanged and `push` returns `true`.

**Parameters:**

`component` - The `java.awt.Component` to be moved.

**Returns:**

returns `true` on success, `false` on failure, for example if the `java.awt.Component` has yet to be added to the `HComponentOrdering`.

---

### **pushBehind(Component, Component)**

```
public boolean pushBehind(java.awt.Component move, java.awt.Component front)
```

Puts the specified `java.awt.Component` behind another `java.awt.Component` in the Z-order of this `HComponentOrdering`.

If `move` and `front` are the same component which has been added to the container `pushBehind` does not change the Z-order and returns `true`.

**Parameters:**

`move` - The `java.awt.Component` to be moved directly behind the "front" Component in the Z-order of this `HComponentOrdering`.

`front` - The `java.awt.Component` which the "move" Component should be placed directly behind.

**Returns:**

returns `true` on success, `false` on failure, for example when either `java.awt.Component` has yet to be added to the `HComponentOrdering`.

---

### **pushToBack(Component)**

```
public boolean pushToBack(java.awt.Component component)
```

Place the specified `java.awt.Component` at the "back" of the Z-order in this `HComponentOrdering`.

If component is already at the back the Z-order is unchanged and `pushToBack` returns `true`.

**Parameters:**

`component` - The `java.awt.Component` to place at the "back" of the Z-order of this `HComponentOrdering`.

**Returns:**

returns `true` on success, `false` on failure, for example when the `java.awt.Component` has yet to be added to the `HComponentOrdering`. If the component was not added to the container `pushToBack` does not change the Z-order.

# org.havi.ui HConfigurationException

## Syntax

```
public class HConfigurationException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--org.havi.ui.HConfigurationException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Thrown when an application requests an `HScreenConfiguration` that cannot be satisfied -- either because the `HScreenConfiguration` does not have the functionality, or because the requested `HScreenConfiguration` is otherwise invalid, e.g. it is an `HScreenConfiguration` due to a different `HScreenDevice` than the one it is being applied to. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
message	Message explaining why the exception was thrown	null	-	java.lang.Throwable#getMessage

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Constructors

### HConfigurationException()

```
public HConfigurationException()
```

Creates an `HConfigurationException` object. See the class description for details of constructor parameters and default values.

### HConfigurationException(String)

```
public HConfigurationException(java.lang.String message)
```

Creates an HConfigurationException object with a specified reason string.

**Parameters:**

`message` - the reason why the exception was raised

# org.havi.ui HContainer

## Syntax

```
public class HContainer extends java.awt.Container implements HMatteLayer, HComponentOrdering,
    TestOpacity
```

```
java.lang.Object
|
+--java.awt.Component
    |
    +--java.awt.Container
        |
        +--org.havi.ui.HContainer
```

## All Implemented Interfaces:

`HComponentOrdering`, `HMatteLayer`, `java.awt.image.ImageObserver`, `java.awt.MenuContainer`, `java.io.Serializable`, `TestOpacity`

## Description

The `HContainer` class extends the `java.awt.Container` class by implementing the `HMatteLayer` interface and providing additional Z-ordering capabilities, which are required since components in the HAVi user-interface are explicitly allowed to overlap each other.

Note that these Z-ordering capabilities (`addBefore`, `addAfter`, `pop`, `popInFrontOf`, `popToFront`, `push`, `pushBehind` and `pushToBack`) must be implemented by (implicitly) reordering the child Components within the `HContainer`, so that the standard AWT convention that the Z-order is defined as the order in which Components are added to a given Container is maintained. For example, one implementation of `popToFront` might be to make the specified Component become the first Component added to the parent Container by removing all Components from that Container, adding the specified Container first, and then adding the remaining Components in their current relative order to that Container. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
width	width of this component in pixels (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
height	height of this component in pixels (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>



Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte ( <code>HMatte</code> ).	none (i.e. <code>getMatte()</code> returns null)	<code>setMatte (HMatte)</code>	<code>getMatte ()</code>
LayoutManager	null (in contrast to <code>java.awt.Container</code> )	<code>java.awt.Container#setLayout</code>	<code>java.awt.Container#getLayout</code>

## Constructors

### `HContainer()`

```
public HContainer()
```

Creates an `HContainer` object. See the class description for details of constructor parameters and default values.

### `HContainer(int, int, int, int)`

```
public HContainer(int x, int y, int width, int height)
```

Creates an `HContainer` object. See the class description for details of constructor parameters and default values.

## Methods

### `addAfter(Component, Component)`

```
public java.awt.Component addAfter(java.awt.Component component, java.awt.Component front)
```

Adds a `java.awt.Component` to this `HContainer` directly behind a previously added `java.awt.Component`.

If `component` has already been added to this container, then `addAfter` moves `component` behind `front`. If `front` and `component` are the same component which was already added to this container, `addAfter` does not change the ordering of the components and returns `component`.

This method affects the Z-order of the `java.awt.Component` children within the `HContainer`, and may also implicitly change the numeric ordering of those children.

#### Specified By:

`addAfter(Component, Component)` in interface `HComponentOrdering`

#### Parameters:

`component` - is the `java.awt.Component` to be added to the `HContainer`

`front` - is the `java.awt.Component`, which `component` will be placed behind, i.e. `front` will be directly in front of the added `java.awt.Component`

#### Returns:

If the `java.awt.Component` is successfully added, then it will be returned from this call. If the `java.awt.Component` is not successfully added, e.g. `front` is not a `java.awt.Component` currently added to the `HContainer`, then `null` will be returned.

This method must be implemented in a thread safe manner.

---

### **addBefore(Component, Component)**

```
public java.awt.Component addBefore(java.awt.Component component,  
    java.awt.Component behind)
```

Adds a `java.awt.Component` to this `HContainer` directly in front of a previously added `java.awt.Component`.

If `component` has already been added to this container, then `addBefore` moves `component` in front of `behind`. If `behind` and `component` are the same component which was already added to this container, `addBefore` does not change the ordering of the components and returns `component`.

This method affects the Z-order of the `java.awt.Component` children within the `HContainer`, and may also implicitly change the numeric ordering of those children.

#### **Specified By:**

`addBefore(Component, Component)` in interface `HComponentOrdering`

#### **Parameters:**

`component` - is the `java.awt.Component` to be added to the `HContainer`

`behind` - is the `java.awt.Component`, which `component` will be placed in front of, i.e. `behind` will be directly behind the added `java.awt.Component`

#### **Returns:**

If the `java.awt.Component` is successfully added, then it will be returned from this call. If the `java.awt.Component` is not successfully added, e.g. `behind` is not a `java.awt.Component` currently added to the `HContainer`, then `null` will be returned.

This method must be implemented in a thread safe manner.

---

### **getMatte()**

```
public HMatte getMatte()
```

Get any `HMatte` currently associated with this component.

#### **Specified By:**

`getMatte()` in interface `HMatteLayer`

#### **Returns:**

the `HMatte` currently associated with this component or `null` if there is no associated matte.

---

### **group()**

```
public void group()
```

Groups the `HContainer` and its components. If the container is already grouped this method has no effect

#### **See Also:**

`ungroup()`, `isGrouped()`

---

### **isDoubleBuffered()**

```
public boolean isDoubleBuffered()
```

Returns `true` if all the drawing done during the update and paint methods for this specific `HContainer` object is automatically double buffered.

**Overrides:**

`java.awt.Component.isDoubleBuffered()` in class `java.awt.Component`

**Returns:**

`true` if all the drawing done during the update and paint methods for this specific `HComponent` object is automatically double buffered, or `false` if drawing is not double buffered. The default value for the double buffering setting is platform-specific.

**isGrouped()**

```
public boolean isGrouped()
```

Tests whether the `HContainer` and its components are grouped. By default the container is not grouped with its components.

**Returns:**

returns `true` if the `HContainer` and its components are grouped, `false` otherwise.

**See Also:**

`group()`, `ungroup()`

**isOpaque()**

```
public boolean isOpaque()
```

Returns `true` if the entire `HContainer` area, as given by the `java.awt.Component#getBounds` method, is fully opaque, i.e. its paint method (or surrogate methods) guarantee that all pixels are painted in an opaque `Color`.

By default, the return value is `false`. The return value should be overridden by subclasses that can guarantee full opacity. The consequences of an invalid overridden value are implementation specific.

**Specified By:**

`isOpaque()` in interface `TestOpacity`

**Returns:**

`true` if all the pixels within the area given by the `java.awt.Component#getBounds` method are fully opaque, i.e. its paint method (or surrogate methods) guarantee that all pixels are painted in an opaque `Color`, otherwise `false`.

**pop(Component)**

```
public boolean pop(java.awt.Component component)
```

Moves the specified `java.awt.Component` one component nearer in the Z-order, i.e. swapping it with the `java.awt.Component` that was directly in front of it.

If `component` is already at the front of the Z-order, the order is unchanged and `pop` returns `true`.

**Specified By:**

`pop(Component)` in interface `HComponentOrdering`

**Parameters:**

`component` - The `java.awt.Component` to be moved.

**Returns:**

returns `true` on success, `false` on failure, for example if the `java.awt.Component` has yet to be added to the `HContainer`.

---

**popInFrontOf(Component, Component)**

```
public boolean popInFrontOf(java.awt.Component move, java.awt.Component behind)
```

Puts the specified `java.awt.Component` in front of another `java.awt.Component` in the Z-order of this `HContainer`.

If `move` and `behind` are the same component which has been added to the container `popInFront` does not change the Z-order and returns `true`.

**Specified By:**

`popInFrontOf(Component, Component)` in interface `HComponentOrdering`

**Parameters:**

`move` - The `java.awt.Component` to be moved directly in front of the "behind" Component in the Z-order of this `HContainer`.

`behind` - The `java.awt.Component` which the "move" Component should be placed directly in front of.

**Returns:**

returns `true` on success, `false` on failure, for example when either `java.awt.Component` has yet to be added to the `HContainer`. If this method fails, the Z-order is unchanged.

---

**popToFront(Component)**

```
public boolean popToFront(java.awt.Component component)
```

Brings the specified `java.awt.Component` to the "front" of the Z-order in this `HContainer`.

If `component` is already at the front of the Z-order, the order is unchanged and `popToFront` returns `true`.

**Specified By:**

`popToFront(Component)` in interface `HComponentOrdering`

**Parameters:**

`component` - The `java.awt.Component` to bring to the "front" of the Z-order of this `HContainer`.

**Returns:**

returns `true` on success, `false` on failure, for example when the `java.awt.Component` has yet to be added to the `HContainer`. If this method fails, the Z-order is unchanged.

---

**push(Component)**

```
public boolean push(java.awt.Component component)
```

Moves the specified `java.awt.Component` one component further away in the Z-order, i.e. swapping it with the `java.awt.Component` that was directly behind it.

If `component` is already at the back of the Z-order, the order is unchanged and `push` returns `true`.

**Specified By:**

`push(Component)` in interface `HComponentOrdering`

**Parameters:**

`component` - The `java.awt.Component` to be moved.

**Returns:**

returns `true` on success, `false` on failure, for example if the `java.awt.Component` has yet to be added to the `HContainer`.

---

**pushBehind(Component, Component)**

```
public boolean pushBehind(java.awt.Component move, java.awt.Component front)
```

Puts the specified `java.awt.Component` behind another `java.awt.Component` in the Z-order of this `HContainer`.

If `move` and `front` are the same component which has been added to the container `pushBehind` does not change the Z-order and returns `true`.

**Specified By:**

`pushBehind(Component, Component)` in interface `HComponentOrdering`

**Parameters:**

`move` - The `java.awt.Component` to be moved directly behind the "front" Component in the Z-order of this `HContainer`.

`front` - The `java.awt.Component` which the "move" Component should be placed directly behind.

**Returns:**

returns `true` on success, `false` on failure, for example when either `java.awt.Component` has yet to be added to the `HContainer`.

---

**pushToBack(Component)**

```
public boolean pushToBack(java.awt.Component component)
```

Place the specified `java.awt.Component` at the "back" of the Z-order in this `HContainer`.

If component is already at the back the Z-order is unchanged and `pushToBack` returns `true`.

**Specified By:**

`pushToBack(Component)` in interface `HComponentOrdering`

**Parameters:**

`component` - The `java.awt.Component` to place at the "back" of the Z-order of this `HContainer`.

**Returns:**

returns `true` on success, `false` on failure, for example when the `java.awt.Component` has yet to be added to the `HContainer`. If the component was not added to the container `pushToBack` does not change the Z-order.

---

**setMatte(HMatte)**

```
public void setMatte(HMatte m)
```

Applies an `HMatte` to this component, for matte compositing. Any existing animated matte must be stopped before this method is called or an `HMatteException` will be thrown.

**Specified By:**

`setMatte(HMatte)` in interface `HMatteLayer`

**Parameters:**

`m` - The `HMatte` to be applied to this component -- note that only one matte may be associated with the component, thus any previous matte will be replaced. If `m` is null, then any matte associated with the component is removed and further calls to `getMatte()` shall return null. The component shall behave as if it had a fully opaque `HFlatMatte` associated with it (i.e an `HFlatMatte` with the default value of 1.0.)

**Throws:**

`HMatteException` - if the `HMatte` cannot be associated with the component. This can occur:

- if the specific matte type is not supported
- if the platform does not support any matte type
- if the component is associated with an already running `HFlatEffectMatte` or `HImageEffectMatte`. The exception is thrown even if `m` is null.

**See Also:**`HMatte`

---

**ungroup()**

```
public void ungroup()
```

Ungroups the `HContainer` and its components. If the container is already ungrouped, this method has no effect.

**See Also:**`group()`, `isGrouped()`

# org.havi.ui HDefaultTextLayoutManager

## Syntax

```
public class HDefaultTextLayoutManager implements HTextLayoutManager
```

```
java.lang.Object
|
+--org.havi.ui.HDefaultTextLayoutManager
```

## All Implemented Interfaces:

```
HTextLayoutManager
```

## Description

The `HDefaultTextLayoutManager` provides the default text rendering mechanism for the `HStaticText`, `HText` and `HTextButton` classes.

The `HDefaultTextLayoutManager` handles alignment and justification of text in both horizontal and vertical directions as specified by the current alignment modes set on `HVisible`. It does not support scaling of text content, and the scaling mode of an associated `HVisible` is ignored.

The string passed to the `render(String, Graphics, HVisible, Insets)` method may be multi-line, where each line is separated by a "\n" (0x0A). If the string does not fit in the space available, the string shall be truncated and an ellipsis ("...") appended to indicate the truncation.

The `HDefaultTextLayoutManager` should query the `HVisible` passed to its `render(String, Graphics, HVisible, Insets)` method to determine the basic font to render text in. If the specified font cannot be accessed the default behavior is to replace it with the nearest builtin font. Each missing character is replaced with an "!" character.

The antialiasing behavior of `HDefaultTextLayoutManager` is platform dependent. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

## See Also:

`HTextLayoutManager`, `HStaticText`, `HText`, `HTextButton`, `HTextLook`, `HVisible`

## Constructors

---

## HDefaultTextLayoutManager()

```
public HDefaultTextLayoutManager()
```

Creates an `HDefaultTextLayoutManager` object. See the class description for details of constructor parameters and default values.

## Methods

---

### render(String, Graphics, HVisible, Insets)

```
public void render(java.lang.String markedUpString, java.awt.Graphics g, HVisible v,  
                  java.awt.Insets insets)
```

Render the string. The `HTextLayoutManager` should use the passed `HVisible` object to determine any additional information required to render the string, e.g. `Font`, `Color` etc.

The text should be laid out in the layout area, which is defined by the bounds of the specified `HVisible`, after subtracting the insets. If the insets are `null` the full bounding rectangle is used as the area to render text into.

The `HTextLayoutManager` should not modify the clipping rectangle of the `Graphics` object.

#### Specified By:

`render(String, Graphics, HVisible, Insets)` in interface `HTextLayoutManager`

#### Parameters:

`markedUpString` - the string to render.

`g` - the graphics context, including a clipping rectangle which encapsulates the area within which rendering is permitted. If a valid insets value is passed to this method then text must only be rendered into the bounds of the widget after the insets are subtracted. If the insets value is `null` then text is rendered into the entire bounding area of the `HVisible`. It is implementation specific whether or not the renderer takes into account the intersection of the clipping rectangle in each case for optimization purposes.

`v` - the `HVisible` into which to render.

`insets` - the insets to determine the area in which to layout the text, or `null`.



# org.havi.ui HEmulatedGraphicsConfiguration

## Syntax

```
public class HEmulatedGraphicsConfiguration extends HGraphicsConfiguration
```

```
java.lang.Object
|
+--HScreenConfiguration
   |
   +--HGraphicsConfiguration
      |
      +--org.havi.ui.HEmulatedGraphicsConfiguration
```

## Description

An `HEmulatedGraphicsConfiguration` is a configuration for a "virtual" graphics device that may perform one or more emulations, e.g. in the ATSC context an `HEmulatedGraphicsDevice` might implement multiple `HEmulatedGraphicsConfiguration`, corresponding to each of the possible relationships to the high-definition display modes. The `HEmulatedGraphicsConfiguration` would be used to configure a device appropriately for rendering into, whilst mapping the emulated device onto the "true" physical display, e.g. by down-sampling to standard-definition display.

In essence the `HEmulatedGraphicsConfiguration` may be considered as a pair of `HGraphicsConfiguration` objects: one describing the configuration of the emulation and the second describing the corresponding configuration of the implementation.

Hence, an `HGraphicsConfiguration` may be considered as a special case of the `HEmulatedGraphicsConfiguration` class, where the emulation and implementation are equivalent. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## See Also:

`HEmulatedGraphicsDevice`, `HGraphicsConfiguration`

## Constructors

### HEmulatedGraphicsConfiguration()

```
protected HEmulatedGraphicsConfiguration()
```

It is not intended that applications should directly construct `HEmulatedGraphicsConfiguration` objects.

Creates an `HEmulatedGraphicsConfiguration` object. See the class description for details of constructor parameters and default values.

## Methods

---

### `getConfigTemplate()`

```
public HGraphicsConfigTemplate getConfigTemplate()
```

Returns an `HGraphicsConfigTemplate` describing the virtual (emulation) characteristics of the `HEmulatedGraphicsDevice`.

Overridden method from `HGraphicsConfiguration` -- for an `HEmulatedGraphicsConfiguration` this returns a description of the emulation characteristics.

**Overrides:**

`getConfigTemplate()` in class `HGraphicsConfiguration`

**Returns:**

an `HGraphicsConfigTemplate` describing the virtual (emulation) characteristics of the `HEmulatedGraphicsDevice`.

**See Also:**

`HGraphicsConfigTemplate`, `HGraphicsConfiguration`, `HEmulatedGraphicsDevice`

---

### `getEmulation()`

```
public HGraphicsConfigTemplate getEmulation()
```

Returns an `HGraphicsConfigTemplate` describing the virtual (emulation) characteristics of the `HEmulatedGraphicsDevice`.

**Returns:**

an `HGraphicsConfigTemplate` describing the virtual (emulation) characteristics of the `HEmulatedGraphicsDevice`.

**See Also:**

`HGraphicsConfigTemplate`, `HEmulatedGraphicsDevice`

---

### `getImplementation()`

```
public HGraphicsConfigTemplate getImplementation()
```

Returns an `HGraphicsConfigTemplate` describing the physical (implementation) characteristics of the `HEmulatedGraphicsDevice`.

**Returns:**

an `HGraphicsConfigTemplate` describing the physical (implementation) characteristics of the `HEmulatedGraphicsDevice`.

**See Also:**

`HGraphicsConfigTemplate`, `HEmulatedGraphicsDevice`

# org.havi.ui HEmulatedGraphicsDevice

## Syntax

```
public abstract class HEmulatedGraphicsDevice extends HGraphicsDevice
```

```
java.lang.Object
|
+--HScreenDevice
|
+--HGraphicsDevice
|
+--org.havi.ui.HEmulatedGraphicsDevice
```

## All Implemented Interfaces:

org.davic.resources.ResourceProxy, org.davic.resources.ResourceServer

## Description

An `HEmulatedGraphicsDevice` is a "virtual" graphics device that has the capability to be configured to perform one (of many) possible emulations. For example, in the DVB context a 4:3 television might have an `HEmulatedGraphicsDevice` that had an `HEmulatedGraphicsConfiguration` that emulated a virtual 14:9 display. The 14:9 `HEmulatedGraphicsConfiguration` would be used for rendering into from AWT, whilst being displayed on the "true" 4:3 physical display. The relationship between the emulation and implementation is encapsulated within the `HEmulatedGraphicsConfiguration`.

An `HEmulatedGraphicsDevice` transforms both AWT pixel-oriented drawing operations and AWT user-input event coordinates, this is performed outside of the Java application (typically in hardware).

An `HEmulatedGraphicsDevice` may (of necessity) modify coordinates for Components and/or events to the nearest physical / virtual pixel --- authors should not depend on single pixel accuracy.

There is no difference to a Java application between an `HGraphicsDevice` and an `HEmulatedGraphicsDevice`, except for the implication of possible rounding errors in integer pixel positions, e.g. Component placement and/or resolution of events.

Java2D mechanisms should behave as per their normal semantics, with respect to display on-screen.

## Constructors

---

### HEmulatedGraphicsDevice()

```
protected HEmulatedGraphicsDevice()
```

It is not intended that applications should directly construct `HEmulatedGraphicsDevice` objects.

Creates an `HEmulatedGraphicsDevice` object. See the class description for details of constructor parameters and default values.

## Methods

---

**setGraphicsConfiguration(HEmulatedGraphicsConfiguration)**

```
public boolean setGraphicsConfiguration(HEmulatedGraphicsConfiguration hegc)
```

Set the graphics configuration for the device.

**Parameters:**

hegc - the `HEmulatedGraphicsConfiguration` to which this device should be set.

**Returns:**

A boolean indicating whether the configuration could be applied successfully. If the configuration could not be applied successfully, the configuration after this method may not match the configuration of the device prior to this method being called --- applications should take steps to determine whether a partial change of settings has been made.

**Throws:**

`SecurityException` - if the application does not have sufficient rights to set the configuration for this device.

`HPermissionDeniedException` - (`HPermissionDeniedException`) if the application does not currently have the right to set the configuration for this device.

`HConfigurationException` - (`HConfigurationException`) if the specified configuration is not valid for this device.

# org.havi.ui

# HEventMulticaster

## Syntax

```
public class HEventMulticaster implements HBackgroundImageListener,
    HScreenConfigurationListener, HScreenLocationModifiedListener,
    java.awt.event.WindowListener, HActionListener, HAdjustmentListener, HFocusListener,
    HItemListener, HTextListener, HKeyListener, org.davic.resources.ResourceStatusListener

java.lang.Object
|
+--org.havi.ui.HEventMulticaster
```

## All Implemented Interfaces:

```
java.awt.event.ActionListener, java.util.EventListener, java.awt.event.FocusListener,
HActionListener, HAdjustmentListener, HBackgroundImageListener,
HFocusListener, HItemListener, HKeyListener, HScreenConfigurationListener,
HScreenLocationModifiedListener, HTextListener, java.awt.event.KeyListener,
org.davic.resources.ResourceStatusListener, java.awt.event.WindowListener
```

## Description

The `HEventMulticaster` class is intended to handle event dispatching for the following HAVi events:

- `HBackgroundImageEvent`
- `HScreenConfigurationEvent`
- `HScreenLocationModifiedEvent`
- `HActionEvent`
- `HFocusEvent`
- `HItemEvent`
- `HTextEvent`
- `HKeyEvent`
- `HAdjustmentEvent`
- `java.awt.event.WindowEvent`
- `org.davic.resources.ResourceStatusEvent`

It is an implementation option for this class to insert other classes in the inheritance tree (for example `java.awt.AWTEventMulticaster`). It is allowed that this may result in `HEventMulticaster` inheriting additional methods beyond those specified here.

Note: the `org.davic.resources.ResourceStatusListener` specification does not require `EventListener` to be present. In a HAVi UI implementation, `ResourceStatusListener` shall extend `EventListener`.

The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

None.

### See Also:

HBackgroundImageListener, HScreenConfigurationListener,  
HScreenLocationModifiedListener, HActionListener, HAdjustmentListener,  
HFocusListener, HItemListener, HTextListener, HKeyListener,  
java.awt.event.WindowListener, org.davic.resources.ResourceStatusListener

## Fields

---

**a**

protected final java.util.EventListener a

---

**b**

protected final java.util.EventListener b

## Constructors

---

### HEventMulticaster(EventListener, EventListener)

protected HEventMulticaster(java.util.EventListener a, java.util.EventListener b)

Creates an event multicaster instance which chains listener-a with listener-b.

**Parameters:**

a - listener-a

b - listener-b

## Methods

---

### actionPerformed(ActionEvent)

public void actionPerformed(java.awt.event.ActionEvent e)

Handles the ActionEvent by invoking the actionPerformed methods on listener-a and listener-b.

**Specified By:**

java.awt.event.ActionListener.actionPerformed(java.awt.event.ActionEvent) in interface  
java.awt.event.ActionListener

**Parameters:**

e - the ActionEvent event

---

**add(HActionListener, HActionListener)**

```
public static HActionListener add(HActionListener a, HActionListener b)
```

Adds `HActionListener -a` with `HActionListener -b` and returns the resulting multicast listener.

**Parameters:**

a - `HActionListener-a`

b - `HActionListener-b`

---

**add(HAdjustmentListener, HAdjustmentListener)**

```
public static HAdjustmentListener add(HAdjustmentListener a, HAdjustmentListener b)
```

Adds `HAdjustmentListener -a` with `HAdjustmentListener -b` and returns the resulting multicast listener.

**Parameters:**

a - `HAdjustmentListener-a`

b - `HAdjustmentListener-b`

---

**add(HBackgroundImageListener, HBackgroundImageListener)**

```
public static HBackgroundImageListener add(HBackgroundImageListener a,  
                                           HBackgroundImageListener b)
```

Adds `HBackgroundImageListener -a` with `HBackgroundImageListener -b` and returns the resulting multicast listener.

**Parameters:**

a - `HBackgroundImageListener-a`

b - `HBackgroundImageListener-b`

---

**add(HFocusListener, HFocusListener)**

```
public static HFocusListener add(HFocusListener a, HFocusListener b)
```

Adds `HFocusListener -a` with `HFocusListener -b` and returns the resulting multicast listener.

**Parameters:**

a - `HFocusListener-a`

b - `HFocusListener-b`

---

**add(HItemListener, HItemListener)**

```
public static HItemListener add(HItemListener a, HItemListener b)
```

Adds `HItemListener -a` with `HItemListener -b` and returns the resulting multicast listener.

**Parameters:**

a - `HItemListener-a`

b - `HItemListener-b`

---

**add(HKeyListener, HKeyListener)**

```
public static HKeyListener add(HKeyListener a, HKeyListener b)
```

Adds `HKeyListener -a` with `HKeyListener -b` and returns the resulting multicast listener.

**Parameters:**

- a - HKeyListener-a
- b - HKeyListener-b

**add(HScreenConfigurationListener, HScreenConfigurationListener)**

```
public static HScreenConfigurationListener add(HScreenConfigurationListener a,
                                             HScreenConfigurationListener b)
```

Adds `HScreenConfigurationListener -a` with `HScreenConfigurationListener -b` and returns the resulting multicast listener.

**Parameters:**

- a - HScreenConfigurationListener-a
- b - HScreenConfigurationListener-b

**add(HScreenConfigurationListener, HScreenConfigurationListener, HScreenConfigTemplate)**

```
public static HScreenConfigurationListener add(HScreenConfigurationListener a,
                                             HScreenConfigurationListener b, HScreenConfigTemplate tb)
```

Adds `HScreenConfigurationListener -a` with `HScreenConfigurationListener -b`, which is notified when the `HScreenDevice` configuration is modified so that it is no longer compatible with the `HScreenConfigTemplate` `tb`. It returns the resulting multicast listener.

**Parameters:**

- a - HScreenConfigurationListener-a
- b - HScreenConfigurationListener-b
- tb - HScreenConfigTemplate associated with HScreenConfigurationListener-b

**add(HScreenLocationModifiedListener, HScreenLocationModifiedListener)**

```
public static HScreenLocationModifiedListener add(HScreenLocationModifiedListener a,
                                                HScreenLocationModifiedListener b)
```

Adds `HScreenLocationModifiedListener -a` with `HScreenLocationModifiedListener -b` and returns the resulting multicast listener.

**Parameters:**

- a - HScreenLocationModifiedListener-a
- b - HScreenLocationModifiedListener-b

**add(HTextListener, HTextListener)**

```
public static HTextListener add(HTextListener a, HTextListener b)
```

Adds `HTextListener -a` with `HTextListener -b` and returns the resulting multicast listener.

**Parameters:**

- a - HTextListener-a
- b - HTextListener-b

**add(ResourceStatusListener, ResourceStatusListener)**

```
public static org.davic.resources.ResourceStatusListener
    add(org.davic.resources.ResourceStatusListener a,
        org.davic.resources.ResourceStatusListener b)
```



Adds ResourceStatusListener-a with listener-b and returns the resulting multicast listener. In a HAVi UI implementation, ResourceStatusListener shall extend EventListener.

**Parameters:**

a - listener-a

b - listener-b

**add(WindowListener, WindowListener)**

```
public static java.awt.event.WindowListener add(java.awt.event.WindowListener a,
        java.awt.event.WindowListener b)
```

Adds WindowListener-a with WindowListener-b and returns the resulting multicast listener.

**Parameters:**

a - WindowListener-a

b - WindowListener-b

**addInternal(EventListener, EventListener)**

```
protected static java.util.EventListener addInternal(java.util.EventListener a,
        java.util.EventListener b)
```

Returns the resulting multicast listener from adding listener-a and listener-b together. If listener-a is null, it returns listener-b; if listener-b is null, it returns listener-a. If neither are null, then it creates and returns a new HEventMulticaster instance which chains a with b.

**Parameters:**

a - event listener-a

b - event listener-b

**caretMoved(HTextEvent)**

```
public void caretMoved(HTextEvent e)
```

Handles the HTextEvent by invoking the caretMoved(HTextEvent) methods on listener-a and listener-b.

**Specified By:**

caretMoved(HTextEvent) in interface HTextListener

**Parameters:**

e - the HTextEvent event

**currentItemChanged(HItemEvent)**

```
public void currentItemChanged(HItemEvent e)
```

Handles the HItemEvent by invoking the currentItemChanged(HItemEvent) methods on listener-a and listener-b.

**Specified By:**

currentItemChanged(HItemEvent) in interface HItemListener

**Parameters:**

e - the HItemEvent event

---

**focusGained(FocusEvent)**

```
public void focusGained(java.awt.event.FocusEvent e)
```

Handles the FocusEvent by invoking the focusGained methods on listener-a and listener-b.

**Specified By:**

```
java.awt.event.FocusListener.focusGained(java.awt.event.FocusEvent) in interface  
java.awt.event.FocusListener
```

**Parameters:**

e - the FocusEvent event

---

**focusLost(FocusEvent)**

```
public void focusLost(java.awt.event.FocusEvent e)
```

Handles the FocusEvent by invoking the focusLost methods on listener-a and listener-b.

**Specified By:**

```
java.awt.event.FocusListener.focusLost(java.awt.event.FocusEvent) in interface  
java.awt.event.FocusListener
```

**Parameters:**

e - the FocusEvent event

---

**imageLoaded(HBackgroundImageEvent)**

```
public void imageLoaded(HBackgroundImageEvent e)
```

Handles the `HBackgroundImageEvent` by invoking the `imageLoaded(HBackgroundImageEvent)` methods on listener-a and listener-b.

**Specified By:**

```
imageLoaded(HBackgroundImageEvent) in interface HBackgroundImageListener
```

**Parameters:**

e - the HBackgroundImageEvent event

---

**imageLoadFailed(HBackgroundImageEvent)**

```
public void imageLoadFailed(HBackgroundImageEvent e)
```

Handles the `HBackgroundImageEvent` by invoking the `imageLoadFailed(HBackgroundImageEvent)` methods on listener-a and listener-b.

**Specified By:**

```
imageLoadFailed(HBackgroundImageEvent) in interface  
HBackgroundImageListener
```

**Parameters:**

e - the HBackgroundImageEvent event

---

**keyPressed(KeyEvent)**

```
public void keyPressed(java.awt.event.KeyEvent e)
```

Handles the `HKeyEvent` by invoking the `keyPressed` methods on listener-a and listener-b.

**Specified By:**

```
java.awt.event.KeyListener.keyPressed(java.awt.event.KeyEvent) in interface  
java.awt.event.KeyListener
```

**Parameters:**

e - the HKeyEvent event

---

**keyReleased(KeyEvent)**

```
public void keyReleased(java.awt.event.KeyEvent e)
```

Handles the `HKeyEvent` by invoking the `keyReleased` methods on listener-a and listener-b.

**Specified By:**

`java.awt.event.KeyListener.keyReleased(java.awt.event.KeyEvent)` in interface  
`java.awt.event.KeyListener`

**Parameters:**

e - the HKeyEvent event

---

**keyTyped(KeyEvent)**

```
public void keyTyped(java.awt.event.KeyEvent e)
```

Handles the `HKeyEvent` by invoking the `keyTyped` methods on listener-a and listener-b.

**Specified By:**

`java.awt.event.KeyListener.keyTyped(java.awt.event.KeyEvent)` in interface  
`java.awt.event.KeyListener`

**Parameters:**

e - the HKeyEvent event

---

**remove(EventListener)**

```
protected java.util.EventListener remove(java.util.EventListener old1)
```

Removes a listener from this multicaster and returns the resulting multicast listener.

**Parameters:**

old1 - the listener to be removed

---

**remove(HActionListener, HActionListener)**

```
public static HActionListener remove(HActionListener l, HActionListener old1)
```

Removes the old `HActionListener` from `HActionListener-l` and returns the resulting multicast listener.

**Parameters:**

l - HActionListener-l

old1 - the HActionListener being removed

---

**remove(HAdjustmentListener, HAdjustmentListener)**

```
public static HAdjustmentListener remove(HAdjustmentListener l, HAdjustmentListener old1)
```

Removes the old `HAdjustmentListener` from `HAdjustmentListener-l` and returns the resulting multicast listener.

**Parameters:**

l - HAdjustmentListener-l

old1 - the HAdjustmentListener being removed

---

**remove(HBackgroundImageListener, HBackgroundImageListener)**

```
public static HBackgroundImageListener remove(HBackgroundImageListener l,
                                             HBackgroundImageListener oldl)
```

Removes the old `HBackgroundImageListener` from `HBackgroundImageListener -l` and returns the resulting multicast listener.

**Parameters:**

- `l` - `HBackgroundImageListener-l`
- `oldl` - the `HBackgroundImageListener` being removed

---

**remove(HFocusListener, HFocusListener)**

```
public static HFocusListener remove(HFocusListener l, HFocusListener oldl)
```

Removes the old `HFocusListener` from `HFocusListener -l` and returns the resulting multicast listener.

**Parameters:**

- `l` - `HFocusListener-l`
- `oldl` - the `HFocusListener` being removed

---

**remove(HItemListener, HItemListener)**

```
public static HItemListener remove(HItemListener l, HItemListener oldl)
```

Removes the old `HItemListener` from `HItemListener -l` and returns the resulting multicast listener.

**Parameters:**

- `l` - `HItemListener-l`
- `oldl` - the `HItemListener` being removed

---

**remove(HKeyListener, HKeyListener)**

```
public static HKeyListener remove(HKeyListener l, HKeyListener oldl)
```

Removes the old `HKeyListener` from `HKeyListener -l` and returns the resulting multicast listener.

**Parameters:**

- `l` - `HKeyListener-l`
- `oldl` - the `HKeyListener` being removed

---

**remove(HScreenConfigurationListener, HScreenConfigurationListener)**

```
public static HScreenConfigurationListener remove(HScreenConfigurationListener l,
                                                  HScreenConfigurationListener oldl)
```

Removes the old `HScreenConfigurationListener` from `HScreenConfigurationListener -l` and returns the resulting multicast listener.

**Parameters:**

- `l` - `HScreenConfigurationListener-l`
- `oldl` - the `HScreenConfigurationListener` being removed

---

**remove(HScreenLocationModifiedListener, HScreenLocationModifiedListener)**

```
public static HScreenLocationModifiedListener remove(HScreenLocationModifiedListener l,
    HScreenLocationModifiedListener oldl)
```

Removes the old `HScreenLocationModifiedListener` from `HScreenLocationModifiedListener-l` and returns the resulting multicast listener.

**Parameters:**

- `l` - `HScreenLocationModifiedListener-l`
- `oldl` - the `HScreenLocationModifiedListener` being removed

---

**remove(HTextListener, HTextListener)**

```
public static HTextListener remove(HTextListener l, HTextListener oldl)
```

Removes the old `HTextListener` from `HTextListener-l` and returns the resulting multicast listener.

**Parameters:**

- `l` - `HTextListener-l`
- `oldl` - the `HTextListener` being removed

---

**remove(ResourceStatusListener, ResourceStatusListener)**

```
public static org.davic.resources.ResourceStatusListener
    remove(org.davic.resources.ResourceStatusListener l,
    org.davic.resources.ResourceStatusListener oldl)
```

Removes the old `ResourceStatusListener` from `ResourceStatusListener-l` and returns the resulting multicast listener. In a HAVi UI implementation, `ResourceStatusListener` shall extend `EventListener`.

**Parameters:**

- `l` - `ResourceStatusListener-l`
- `oldl` - the `ResourceStatusListener` being removed

---

**remove(WindowListener, WindowListener)**

```
public static java.awt.event.WindowListener remove(java.awt.event.WindowListener l,
    java.awt.event.WindowListener oldl)
```

Removes the old `WindowListener` from `WindowListener-l` and returns the resulting multicast listener.

**Parameters:**

- `l` - `WindowListener-l`
- `oldl` - the `WindowListener` being removed

---

**removeInternal(EventListener, EventListener)**

```
protected static java.util.EventListener removeInternal(java.util.EventListener l,
    java.util.EventListener oldl)
```

Returns the resulting multicast listener after removing the old listener from `listener-l`. If `listener-l` equals the old listener OR `listener-l` is null, returns null. Else if `listener-l` is an instance of `HEventMulticaster`, then it removes the old listener from it. Else, returns `listener l`.

**Parameters:**

- `l` - the listener being removed from
- `oldl` - the listener being removed

---

**report(HScreenConfigurationEvent)**

```
public void report(HScreenConfigurationEvent e)
```

Handles the `HScreenConfigurationEvent` by invoking the `report(HScreenConfigurationEvent)` methods on listener-a and listener-b.

**Specified By:**

```
report(HScreenConfigurationEvent) in interface HScreenConfigurationListener
```

**Parameters:**

e - the `HScreenConfigurationEvent` event

---

**report(HScreenLocationModifiedEvent)**

```
public void report(HScreenLocationModifiedEvent e)
```

Handles the `HScreenLocationModifiedEvent` by invoking the `report(HScreenLocationModifiedEvent)` methods on listener-a and listener-b.

**Specified By:**

```
report(HScreenLocationModifiedEvent) in interface  
HScreenLocationModifiedListener
```

**Parameters:**

e - the `HScreenLocationModifiedEvent` event

---

**selectionChanged(HItemEvent)**

```
public void selectionChanged(HItemEvent e)
```

Handles the `HItemEvent` by invoking the `selectionChanged(HItemEvent)` methods on listener-a and listener-b.

**Specified By:**

```
selectionChanged(HItemEvent) in interface HItemListener
```

**Parameters:**

e - the `HItemEvent` event

---

**statusChanged(ResourceStatusEvent)**

```
public void statusChanged(org.davic.resources.ResourceStatusEvent e)
```

Handles the `ResourceStatusEvent` by invoking the `statusChanged` methods on listener-a and listener-b.

**Specified By:**

```
org.davic.resources.ResourceStatusListener.statusChanged(org.davic.resources.ResourceStat  
usEvent) in interface org.davic.resources.ResourceStatusListener
```

**Parameters:**

e - the `ResourceStatusEvent` event

---

**textChanged(HTextEvent)**

```
public void textChanged(HTextEvent e)
```

Handles the `HTextEvent` by invoking the `textChanged(HTextEvent)` methods on listener-a and listener-b.

**Specified By:**

`textChanged(HTextEvent)` in interface `HTextListener`

**Parameters:**

e - the `HTextEvent` event

---

**valueChanged(HAdjustmentEvent)**

```
public void valueChanged(HAdjustmentEvent e)
```

Handles the `HAdjustmentEvent` by invoking the `valueChanged(HAdjustmentEvent)` methods on listener-a and listener-b.

**Specified By:**

`valueChanged(HAdjustmentEvent)` in interface `HAdjustmentListener`

**Parameters:**

e - the `HAdjustmentEvent` event

---

**windowActivated(WindowEvent)**

```
public void windowActivated(java.awt.event.WindowEvent e)
```

Handles the `windowActivated` event by invoking the `windowActivated` methods on listener-a and listener-b.

**Specified By:**

`java.awt.event.WindowListener.windowActivated(java.awt.event.WindowEvent)` in interface `java.awt.event.WindowListener`

**Parameters:**

e - the window event

---

**windowClosed(WindowEvent)**

```
public void windowClosed(java.awt.event.WindowEvent e)
```

Handles the `windowClosed` event by invoking the `windowClosed` methods on listener-a and listener-b.

**Specified By:**

`java.awt.event.WindowListener.windowClosed(java.awt.event.WindowEvent)` in interface `java.awt.event.WindowListener`

**Parameters:**

e - the window event

---

**windowClosing(WindowEvent)**

```
public void windowClosing(java.awt.event.WindowEvent e)
```

Handles the `windowClosing` event by invoking the `windowClosing` methods on listener-a and listener-b.

**Specified By:**

`java.awt.event.WindowListener.windowClosing(java.awt.event.WindowEvent)` in interface `java.awt.event.WindowListener`

**Parameters:**

e - the window event

---

**windowDeactivated(WindowEvent)**

```
public void windowDeactivated(java.awt.event.WindowEvent e)
```

Handles the windowDeactivated event by invoking the windowDeactivated methods on listener-a and listener-b.

**Specified By:**

java.awt.event.WindowListener.windowDeactivated(java.awt.event.WindowEvent) in interface java.awt.event.WindowListener

**Parameters:**

e - the window event

---

**windowDeiconified(WindowEvent)**

```
public void windowDeiconified(java.awt.event.WindowEvent e)
```

Handles the windowDeiconified event by invoking the windowDeiconified methods on listener-a and listener-b.

**Specified By:**

java.awt.event.WindowListener.windowDeiconified(java.awt.event.WindowEvent) in interface java.awt.event.WindowListener

**Parameters:**

e - the window event

---

**windowIconified(WindowEvent)**

```
public void windowIconified(java.awt.event.WindowEvent e)
```

Handles the windowIconified event by invoking the windowIconified methods on listener-a and listener-b.

**Specified By:**

java.awt.event.WindowListener.windowIconified(java.awt.event.WindowEvent) in interface java.awt.event.WindowListener

**Parameters:**

e - the window event

---

**windowOpened(WindowEvent)**

```
public void windowOpened(java.awt.event.WindowEvent e)
```

Handles the windowOpened event by invoking the windowOpened methods on listener-a and listener-b.

**Specified By:**

java.awt.event.WindowListener.windowOpened(java.awt.event.WindowEvent) in interface java.awt.event.WindowListener

**Parameters:**

e - the window event



# org.havi.ui HFlatEffectMatte

## Syntax

```
public class HFlatEffectMatte implements HMatte, HAnimateEffect
```

```
java.lang.Object
|
+--org.havi.ui.HFlatEffectMatte
```

## All Implemented Interfaces:

```
HAnimateEffect, HMatte
```

## Description

The `HFlatEffectMatte` class represents a matte that is constant over space but varies over time. It is specified as a sequence of floating point values in the range 0.0 to 1.0 where:

- 0.0 is fully transparent
- values between 0.0 and 1.0 are partially transparent to the nearest supported transparency value.
- 1.0 is fully opaque

The data for any `HFlatEffectMatte` may be changed "on the fly" using the `setMatteData(float[])` method. However, some implementations may be asynchronously referencing their content (i.e. through a separate implementation-specific animation thread). Therefore the following restrictions apply to the `setMatteData(float[])` method:

- The method must be synchronized with any implementation-specific animation thread such that content cannot be changed while a different thread is using it.
- If the animation was running the method should stop the animation in a synchronized manner before changing content.
- The method should reset the animation to a starting position defined by the current play mode. The repeat count of the animation should be reset to 0.
- If the animation was running the method should start the animation.

The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
data	The transparency value for this flat effect matte.	null (the matte should be treated as being temporally unvarying and opaque)	<code>setMatteData(float[])</code>	<code>getMatteData()</code>

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
The initial piece of content to be presented, i.e. its position in the content array.	0	<code>setPosition(int)</code>	<code>getPosition()</code>

By default the animation should be stopped. Hence, to start the animation its start method must be explicitly invoked. This mechanism allows for animations that are programmatically controlled, e.g. via the setPosition method.

"stopped"

start() / isAnimated()  
stop()

## Constructors

---

### HFlatEffectMatte()

```
public HFlatEffectMatte()
```

Creates an `HFlatEffectMatte` object. See the class description for details of constructor parameters and default values.

---

### HFlatEffectMatte(float[])

```
public HFlatEffectMatte(float[] data)
```

Creates an `HFlatEffectMatte` object. See the class description for details of constructor parameters and default values.

## Methods

---

### getDelay()

```
public int getDelay()
```

Gets the presentation delay for this `HFlatEffectMatte`.

**Specified By:**

`getDelay()` in interface `HAnimateEffect`

**Returns:**

the presentation delay in units of 0.1 seconds.

---

### getMatteData()

```
public float[] getMatteData()
```

Returns the matte data used for this matte.

**Returns:**

the data used for this matte (an array of numbers), or null if no matte data has been set.

---

### getPlayMode()

```
public int getPlayMode()
```

Gets the playing mode for this `HFlatEffectMatte`.

**Specified By:**

`getPlayMode()` in interface `HAnimateEffect`

**Returns:**

the play mode for this `HFlatEffectMatte` .

---

**getPosition()**

```
public int getPosition()
```

Get the current index into the content array which this `HFlatEffectMatte` is using to display content.

**Specified By:**

```
getPosition() in interface HAnimateEffect
```

**Returns:**

the index of the content currently being displayed, in the range  $0 \leq \text{index} < \text{length}$

---

**getRepeatCount()**

```
public int getRepeatCount()
```

Gets the number of times that this `HFlatEffectMatte` is to be played. Note that this method does *not* return the number of repeats that are remaining to be played.

**Specified By:**

```
getRepeatCount() in interface HAnimateEffect
```

**Returns:**

the total number of times that an `HFlatEffectMatte` is to be played. The returned value shall be greater than zero, or `REPEAT_INFINITE` .

---

**isAnimated()**

```
public boolean isAnimated()
```

This method indicates the animation (running) state of the `HFlatEffectMatte` .

**Specified By:**

```
isAnimated() in interface HAnimateEffect
```

**Returns:**

`true` if this `HFlatEffectMatte` is running, i.e. the `start` method has been invoked - `false` otherwise.

---

**setDelay(int)**

```
public void setDelay(int count)
```

Sets the delay between the presentation of successive pieces of content (frames).

After calling `setDelay(int)` on a currently playing `HFlatEffectMatte` , there is no guarantee that one or more frames will not be displayed using the previous delay until the new delay value takes effect.

**Specified By:**

```
setDelay(int) in interface HAnimateEffect
```

**Parameters:**

`count` - the content presentation delay in units of 0.1 seconds duration. If `count` is less than one "unit", then it shall be treated as if it were a delay of one "unit", i.e. 0.1 seconds.

---

**setMatteData(float[])**

```
public void setMatteData(float[] data)
```

Sets the data for this matte. Any previously set data is replaced. If this method is called when the animation is running the data is changed immediately and the current animation position is reset according to the active play mode. The changes affect the animation immediately.

**Parameters:**

`data` - the data for this matte. Specify a null object to remove the associated data for this matte. If the length of the data array is zero, an `IllegalArgumentException` is thrown.

### setPlayMode(int)

```
public void setPlayMode(int mode)
```

Sets the playing mode for this `HFlatEffectMatte`. If the animation is already running a call to `setPlayMode` will change the current value and affect the animation immediately. The position of the animation is unchanged.

**Specified By:**

`setPlayMode(int)` in interface `HAnimateEffect`

**Parameters:**

`mode` - the play mode for this `HFlatEffectMatte`, which must be either `PLAY_ALTERNATING` or `PLAY_REPEATING`.

### setPosition(int)

```
public void setPosition(int position)
```

Set this `HFlatEffectMatte` to display the content at the specified position. If the animation is already running a call to `setPosition` will change the current value and affect the animation immediately.

**Specified By:**

`setPosition(int)` in interface `HAnimateEffect`

**Parameters:**

`position` - an index into the content array which specifies the next piece of content to be displayed. If `position` is less than 0, then the array element at index 0 is displayed, if `position` is greater than or equal to the length of the content array, then the array element at index `[length-1]` will be used.

### setRepeatCount(int)

```
public void setRepeatCount(int count)
```

Sets the number of times that this `HFlatEffectMatte` should be played. If the animation is already running a call to `setRepeatCount` will change the current value and reset the current number of repeats to 0, affecting the animation immediately.

**Specified By:**

`setRepeatCount(int)` in interface `HAnimateEffect`

**Parameters:**

`count` - the number of times that an `HFlatEffectMatte` should be played. Valid values of the repeat count are one or more, and `REPEAT_INFINITE`.

### start()

```
public void start()
```

This method starts this `HFlatEffectMatte` playing. If `start` is called when the animation is already running it resets the animation according to the current play mode, as returned by `getPlayMode()`.

**Specified By:**

`start()` in interface `HAnimateEffect`

---

**stop()**

`public void stop()`

This method indicates that the running `HFlatEffectMatte` should be stopped. After calling this method, there is no guarantee that one or more frames will not be displayed before the animation actually stops playing. If the animation is already stopped further calls to `stop` have no effect.

**Specified By:**

`stop()` in interface `HAnimateEffect`

# org.havi.ui

# HFlatMatte

## Syntax

public class HFlatMatte implements HMatte

```
java.lang.Object
|
+--org.havi.ui.HFlatMatte
```

## All Implemented Interfaces:

HMatte

## Description

The `HFlatMatte` class represents a matte that is constant over space and time. It is specified as a floating point value in the range 0.0 to 1.0 where:

- 0.0 is fully transparent
- values between 0.0 and 1.0 are partially transparent to the nearest supported transparency value.
- 1.0 is fully opaque

The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
data	The transparency value for this flat effect matte.	1.0	<code>setMatteData(float)</code>	<code>getMatteData()</code>

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Constructors

### HFlatMatte()

```
public HFlatMatte()
```

Creates an `HFlatMatte` object. See the class description for details of constructor parameters and default values.

### HFlatMatte(float)

```
public HFlatMatte(float data)
```

Creates an `HFlatMatte` object. See the class description for details of constructor parameters and default values.

## Methods

---

### **getMatteData()**

```
public float getMatteData()
```

Returns the data used for this matte.

**Returns:**

the data used for this matte (a single number).

---

### **setMatteData(float)**

```
public void setMatteData(float data)
```

Sets the data for this matte. Any previously set data is replaced.

**Parameters:**

`data` - the data for this matte.

# org.havi.ui

# HFontCapabilities

## Syntax

```
public class HFontCapabilities
    java.lang.Object
    |
    +--org.havi.ui.HFontCapabilities
```

## Description

The `HFontCapabilities` class allows applications to query the rendering support for various character ranges and individual characters within specified fonts.

## Fields

---

### ALPHABETIC\_PRESENTATION\_FORMS\_A

```
public static final int ALPHABETIC_PRESENTATION_FORMS_A
```

This corresponds to the character range (U+FB00..U+FB4F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

### ARABIC\_EXTENDED

```
public static final int ARABIC_EXTENDED
```

This corresponds to the character range (U+653..U+6FF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

### ARABIC\_PRESENTATION\_FORMS\_A

```
public static final int ARABIC_PRESENTATION_FORMS_A
```

This corresponds to the character range (U+FB50..U+FDFF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

### ARABIC\_PRESENTATION\_FORMS\_B

```
public static final int ARABIC_PRESENTATION_FORMS_B
```

This corresponds to the character range (U+FE70..U+FEFE) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

### ARMENIAN

```
public static final int ARMENIAN
```

This corresponds to the character range (U+530..U+58F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A



---

**ARROWS**

```
public static final int ARROWS
```

This corresponds to the character range (U+2190..U+21FF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**BASIC\_ARABIC**

```
public static final int BASIC_ARABIC
```

This corresponds to the character range (U+600..U+652) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**BASIC\_GEORGIAN**

```
public static final int BASIC_GEORGIAN
```

This corresponds to the character range (U+10D0..U+10FF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**BASIC\_GREEK**

```
public static final int BASIC_GREEK
```

This corresponds to the character range (U+0370..U+03CF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**BASIC\_HEBREW**

```
public static final int BASIC_HEBREW
```

This corresponds to the character range (U+5D0..U+5EA) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**BASIC\_LATIN**

```
public static final int BASIC_LATIN
```

This corresponds to the character range (U+0020..U+007E) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**BENGALI**

```
public static final int BENGALI
```

This corresponds to the character ranges (U+980..U+9FF) and (U+200C..U+200D) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**BLOCK\_ELEMENTS**

```
public static final int BLOCK_ELEMENTS
```

This corresponds to the character range (U+2580..U+259F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**BOPOMOFO**

```
public static final int BOPOMOFO
```

This corresponds to the character range (U+3100..U+312F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**BOX\_DRAWING**

```
public static final int BOX_DRAWING
```

This corresponds to the character range (U+2500..U+257F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**CJK\_COMPATIBILITY**

```
public static final int CJK_COMPATIBILITY
```

This corresponds to the character range (U+3300..U+33FF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**CJK\_COMPATIBILITY\_FORMS**

```
public static final int CJK_COMPATIBILITY_FORMS
```

This corresponds to the character range (U+FE30..U+FE4F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**CJK\_COMPATIBILITY\_IDEOGRAPHS**

```
public static final int CJK_COMPATIBILITY_IDEOGRAPHS
```

This corresponds to the character range (U+F900..U+FAFF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**CJK\_MISCELLANEOUS**

```
public static final int CJK_MISCELLANEOUS
```

This corresponds to the character range (U+3190..U+319F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**CJK\_SYMBOLS\_AND\_PUNCTUATION**

```
public static final int CJK_SYMBOLS_AND_PUNCTUATION
```

This corresponds to the character range (U+3000..U+303F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**CJK\_UNIFIED\_IDEOGRAPHS**

```
public static final int CJK_UNIFIED_IDEOGRAPHS
```

This corresponds to the character range (U+4E00..U+9FFF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**COMBINING\_DIACRITICAL\_MARKS**

```
public static final int COMBINING_DIACRITICAL_MARKS
```

This corresponds to the character range (U+0300..U+036F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**COMBINING\_DIACRITICAL\_MARKS\_FOR\_SYMBOLS**

```
public static final int COMBINING_DIACRITICAL_MARKS_FOR_SYMBOLS
```

This corresponds to the character range (U+20D0..U+20FF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**COMBINING\_HALF\_MARKS**

```
public static final int COMBINING_HALF_MARKS
```

This corresponds to the character range (U+FE20..U+FE2F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**CONTROL\_PICTURES**

```
public static final int CONTROL_PICTURES
```

This corresponds to the character range (U+2400..U+243F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**CURRENCY\_SYMBOLS**

```
public static final int CURRENCY_SYMBOLS
```

This corresponds to the character range (U+20A0..U+20CF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**CYRILLIC**

```
public static final int CYRILLIC
```

This corresponds to the character range (U+400..U+4FF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**DEVANAGARI**

```
public static final int DEVANAGARI
```

This corresponds to the character ranges (U+0900..U+097F) and (U+200C..U+200D) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**DINGBATS**

```
public static final int DINGBATS
```

This corresponds to the character range (U+2700..U+27BF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**ENCLOSED\_ALPHANUMERICS**

public static final int ENCLOSED\_ALPHANUMERICS

This corresponds to the character range (U+2460..U+24FF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**ENCLOSED\_CJK\_LETTERS\_AND\_MONTHS**

public static final int ENCLOSED\_CJK\_LETTERS\_AND\_MONTHS

This corresponds to the character range (U+3200..U+32FF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**GENERAL\_PUNCTUATION**

public static final int GENERAL\_PUNCTUATION

This corresponds to the character range (U+2000..U+206F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**GEOMETRICAL\_SHAPES**

public static final int GEOMETRICAL\_SHAPES

This corresponds to the character range (U+25A0..U+25FF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**GEORGIAN\_EXTENDED**

public static final int GEORGIAN\_EXTENDED

This corresponds to the character range (U+10A0..U+10CF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**GREEK\_EXTENDED**

public static final int GREEK\_EXTENDED

This corresponds to the character range (U+1F00..U+1FFF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**GREEK\_SYMBOLS\_AND\_COPTIC**

public static final int GREEK\_SYMBOLS\_AND\_COPTIC

This corresponds to the character range (U+3D0..U+3FF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**GUJARATI**

public static final int GUJARATI

This corresponds to the character ranges (U+0A80..U+0AFF) and (U+200C..U+200D) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**GURMUKHI**

```
public static final int GURMUKHI
```

This corresponds to the character ranges (U+0A00..U+0A7F) and (U+200C..U+200D) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**HALFWIDTH\_AND\_FULLWIDTH\_FORMS**

```
public static final int HALFWIDTH_AND_FULLWIDTH_FORMS
```

This corresponds to the character range (U+FF00..U+FFEF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**HANGUL**

```
public static final int HANGUL
```

This corresponds to the character range (U+3400..U+3D2D) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**HANGUL\_COMPATIBILITY\_JAMO**

```
public static final int HANGUL_COMPATIBILITY_JAMO
```

This corresponds to the character range (U+3130..U+318F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**HANGUL\_JAMO**

```
public static final int HANGUL_JAMO
```

This corresponds to the character range (U+1100..U+11FF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**HANGUL\_SUPPLEMENTARY\_A**

```
public static final int HANGUL_SUPPLEMENTARY_A
```

This corresponds to the character range (U+3D2E..U+44B7) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**HANGUL\_SUPPLEMENTARY\_B**

```
public static final int HANGUL_SUPPLEMENTARY_B
```

This corresponds to the character range (U+44B8..U+4DFF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**HEBREW\_EXTENDED**

```
public static final int HEBREW_EXTENDED
```

This corresponds to the character ranges (U+590..U+5CF) and (U+5EB..U+5FF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**HIRAGANA**

```
public static final int HIRAGANA
```

This corresponds to the character range (U+3040..U+309F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**IPA\_EXTENSIONS**

```
public static final int IPA_EXTENSIONS
```

This corresponds to the character range (U+0250..U+02AF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**KANNADA**

```
public static final int KANNADA
```

This corresponds to the character ranges (U+0C80..U+0CFF) and (U+200C..U+200D) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**KATAKANA**

```
public static final int KATAKANA
```

This corresponds to the character range (U+30A0..U+30FF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**LAO**

```
public static final int LAO
```

This corresponds to the character range (U+0E80..U+0EFF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**LATIN\_1\_SUPPLEMENT**

```
public static final int LATIN_1_SUPPLEMENT
```

This corresponds to the character range (U+00A0..U+00FF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**LATIN\_EXTENDED\_A**

```
public static final int LATIN_EXTENDED_A
```

This corresponds to the character range (U+0100..U+017F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**LATIN\_EXTENDED\_ADDITIONAL**

```
public static final int LATIN_EXTENDED_ADDITIONAL
```

This corresponds to the character range (U+1E00..U+1EFF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**LATIN\_EXTENDED\_B**

```
public static final int LATIN_EXTENDED_B
```

This corresponds to the character range (U+0180..U+024F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**LETTERLIKE\_SYMBOLS**

```
public static final int LETTERLIKE_SYMBOLS
```

This corresponds to the character range (U+2100..U+214F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**MALAYALAM**

```
public static final int MALAYALAM
```

This corresponds to the character ranges (U+0D00..U+0D7F) and (U+200C..U+200D) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**MATHEMATICAL\_OPERATORS**

```
public static final int MATHEMATICAL_OPERATORS
```

This corresponds to the character range (U+2200..U+22FF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**MISCELLANEOUS\_SYMBOLS**

```
public static final int MISCELLANEOUS_SYMBOLS
```

This corresponds to the character range (U+2600..U+26FF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**MISCELLANEOUS\_TECHNICAL**

```
public static final int MISCELLANEOUS_TECHNICAL
```

This corresponds to the character range (U+2300..U+23FF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**NUMBER\_FORMS**

```
public static final int NUMBER_FORMS
```

This corresponds to the character range (U+2150..U+218F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**OPTICAL\_CHARACTER\_RECOGNITION**

```
public static final int OPTICAL_CHARACTER_RECOGNITION
```

This corresponds to the character range (U+2440..U+245F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**ORIYA**

```
public static final int ORIYA
```

This corresponds to the character ranges (U+0B00..U+0B7F) and (U+200C..U+200D) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**PRIVATE\_USE\_AREA**

```
public static final int PRIVATE_USE_AREA
```

This corresponds to the character range (U+E000..U+F8FF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**SMALL\_FORM\_VARIANTS**

```
public static final int SMALL_FORM_VARIANTS
```

This corresponds to the character range (U+FE50..U+FE6F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**SPACING\_MODIFIER\_LETTERS**

```
public static final int SPACING_MODIFIER_LETTERS
```

This corresponds to the character range (U+02B0..U+02FF) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**SPECIALS**

```
public static final int SPECIALS
```

This corresponds to the character range (U+FFF0..U+FFFD) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**SUPERSCRIPTS\_AND\_SUBSCRIPTS**

```
public static final int SUPERSCRIPTS_AND_SUBSCRIPTS
```

This corresponds to the character range (U+2070..U+209F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**TAMIL**

```
public static final int TAMIL
```

This corresponds to the character ranges (U+0B80..U+0BFF) and (U+200C..U+200D) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

---

**TELUGU**

```
public static final int TELUGU
```

This corresponds to the character ranges (U+0C00..U+0C7F) and (U+200C..U+200D) as defined in ISO/IEC 10646-1:1993(E) normative Annex A



---

## THAI

```
public static final int THAI
```

This corresponds to the character range (U+0E00..U+0E7F) as defined in ISO/IEC 10646-1:1993(E) normative Annex A

## Constructors

---

### HFontCapabilities()

```
protected HFontCapabilities()
```

It is not intended that applications should directly construct `HFontCapabilities` objects.

This method is protected to allow the platform to override it in a different package scope.

## Methods

---

### downloadFont(Font)

```
public static void downloadFont(java.awt.Font f)
```

Download a font which is only temporarily available on a platform. If called for fonts which are permanently available on a platform, this method has no effect. This method blocks while the font is downloaded. If the font is not accessible then this method has no effect and the normal Java font behavior is followed. The platform will use a platform dependent approximation of this font if needed. If a font is already downloaded then it shall not be downloaded again.

**Parameters:**

`f` - the font to load.

**Throws:**

`java.io.IOException` - if an error occurs while trying to download a font.

---

### fontAccessible(Font)

```
public static boolean fontAccessible(java.awt.Font f)
```

Tests if the specified font is accessible in a form usable with this graphics configuration. For platforms which define a mechanism for (possibly temporary) download of fonts, this method shall check fonts available through that mechanism. This does not cause the font to be downloaded but indicates whether a download will succeed. If information needed to determine the existence of the font needs to be downloaded then that information shall be downloaded as part of the execution of this method. This method shall block while any such downloading happens. An error in the downloading of any such information shall be reported as the font not existing.

**Parameters:**

`f` - the font to test.

**Returns:**

`true` if the font is accessible, otherwise `false`.

---

**getSupportedCharacterRanges(Font)**

```
public static int[] getSupportedCharacterRanges(java.awt.Font font)
```

Returns the set of character ranges as defined in ISO/IEC 10646-1:1993(E) normative Annex A that this font supports, or a null array if the capabilities of the font are unknown.

Support for a character range does not imply that ALL characters within that range are available in the specified font.

When deciding whether a particular character range (U+XXXX..U+YYYY) is supported, characters which are in the GENERAL PUNCTUATION range shall not be considered for character ranges other than GENERAL PUNCTUATION.

**Parameters:**

`font` - The font to query for its support for Unicode ranges.

**Returns:**

An array of integer values, as defined in ISO/IEC 10646-1:1993(E) normative Annex A that this font supports, or null.

---

**isCharAvailable(Font, char)**

```
public static boolean isCharAvailable(java.awt.Font font, char c)
```

Returns whether a specific character is available within the specified font, and can be used as defined in ISO/IEC 10646-1:1993(E) specification by the rendering system, e.g. if rendering of bi-directional text, using BI-DIRECTIONAL\_FORMAT\_MARKS is supported

**Parameters:**

`font` - The font to query for its support for the specified character.

`c` - The character whose presence should be tested.

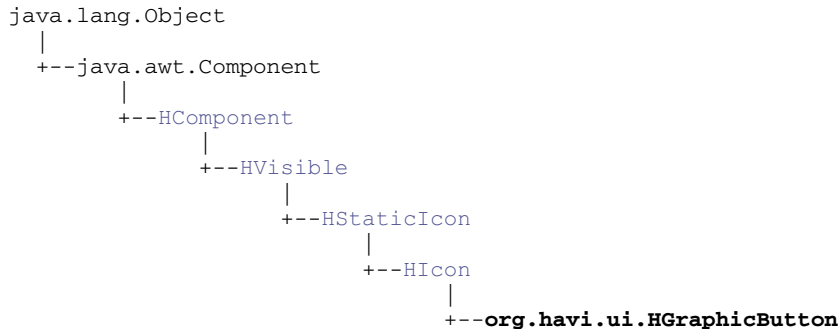
**Returns:**

`true` is the character is available within the font and can be rendered as defined in the ISO/IEC 10646-1:1993(E) specification, `false` otherwise.

# org.havi.ui HGraphicButton

## Syntax

```
public class HGraphicButton extends HIcon implements HActionable
```



## Direct Known Subclasses:

`HToggleButton`

## All Implemented Interfaces:

`HActionable`, `HActionInputPreferred`, `HMatteLayer`, `HNavigable`,  
`HNavigationInputPreferred`, `HNoInputPreferred`, `HState`,  
`java.awt.image.ImageObserver`, `java.awt.MenuContainer`, `java.io.Serializable`, `TestOpacity`

## Description

The `HGraphicButton` is a user interface component representing a push-button with static read-only graphical content. This component can be navigated to, i.e. it can have the input focus, and it can also be actioned as defined by the `HActionable` interface. By default it uses the `HGraphicLook` class to render itself. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
width	width of this component in pixels (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
height	height of this component in pixels (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>

image	The image to be used as the content for every state of this component.	null	setGraphicContent(Image, int)	getGraphicContent(int)
imageNormal	The image to be used as the content for the <code>NORMAL_STATE</code> state of this component.	null	setGraphicContent(Image, int)	getGraphicContent(int)
imageFocus	The image to be used as the content for the focused states of this component.	null	setGraphicContent(Image, int)	getGraphicContent(int)
imageAction	The image to be used as the content for the actioned states of this component.	null	setGraphicContent(Image, int)	getGraphicContent(int)

#### Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte ( <code>HMatte</code> ).	none (i.e. <code>getMatte()</code> returns null)	<code>setMatte(HMatte)</code>	<code>getMatte()</code>
The text layout manager responsible for text formatting.	An <code>HDefaultTextLayoutManager</code> object.	<code>setTextLayoutManager(HTextLayoutManager)</code>	<code>getTextLayoutManager()</code>
The background painting mode	<code>NO_BACKGROUND_FILL</code>	<code>setBackgroundMode(int)</code>	<code>getBackgroundMode()</code>
The default preferred size	not set (i.e. null) unless specified by width and height parameters	<code>setDefaultSize(Dimension)</code>	<code>getDefaultSize()</code>
The horizontal content alignment	<code>HALIGN_CENTER</code>	<code>setHorizontalAlignment(int)</code>	<code>getHorizontalAlignment()</code>
The vertical content alignment	<code>VALIGN_CENTER</code>	<code>setVerticalAlignment(int)</code>	<code>getVerticalAlignment()</code>
The content scaling mode	<code>RESIZE_NONE</code>	<code>setResizeMode(int)</code>	<code>getResizeMode()</code>
The default "look" for this class.	A platform specific <code>HGraphicLook</code>	<code>setDefaultLook(HGraphicLook)</code>	<code>getDefaultLook()</code>
The "look" for this object.	The <code>HGraphicLook</code> returned from <code>HGraphicButton.getDefaultLook</code> when this object was created.	<code>setLook(HLook)</code>	<code>getLook()</code>
The gain focus sound.	null	<code>setGainFocusSound(HSound)</code>	<code>getGainFocusSound()</code>

The lose focus sound.	null	<code>setLoseFocusSound(HSound)</code>	<code>getLoseFocusSound()</code>
The action sound.	null	<code>setActionSound(HSound)</code>	<code>getActionSound()</code>

**See Also:**

`HStaticIcon`, `HIcon`, `HNavigable`, `HActionable`

## Constructors

---

**HGraphicButton()**

```
public HGraphicButton()
```

Creates an `HGraphicButton` object. See the class description for details of constructor parameters and default values.

---

**HGraphicButton(Image)**

```
public HGraphicButton(java.awt.Image image)
```

Creates an `HGraphicButton` object. See the class description for details of constructor parameters and default values.

---

**HGraphicButton(Image, Image, Image)**

```
public HGraphicButton(java.awt.Image imageNormal, java.awt.Image imageFocused,
    java.awt.Image imageActioned)
```

Creates an `HGraphicButton` object. See the class description for details of constructor parameters and default values.

---

**HGraphicButton(Image, Image, Image, int, int, int, int)**

```
public HGraphicButton(java.awt.Image imageNormal, java.awt.Image imageFocused,
    java.awt.Image imageActioned, int x, int y, int width, int height)
```

Creates an `HGraphicButton` object. See the class description for details of constructor parameters and default values.

---

**HGraphicButton(Image, int, int, int, int)**

```
public HGraphicButton(java.awt.Image image, int x, int y, int width, int height)
```

Creates an `HGraphicButton` object. See the class description for details of constructor parameters and default values.

## Methods

---

**addActionListener(HActionListener)**

```
public void addActionListener(HActionListener l)
```

Adds the specified `HActionListener` to receive `HActionEvent` events sent from this `HGraphicButton`. If the listener has already been added further calls will add further references to the listener, which will then receive multiple copies of a single event.

**Specified By:**

`addActionListener(HActionListener)` in interface `HActionable`

**Parameters:**

l - the `HActionListener`.

---

**getActionCommand()**

```
public java.lang.String getActionCommand()
```

Gets the command name for the `HActionEvent` event fired by this `HGraphicButton`.

**Specified By:**

`getActionCommand()` in interface `HActionable`

**Returns:**

A `String` representing the command name of the `HActionEvent` fired by this `HGraphicButton`.

**See Also:**

`getActionCommand()`

---

**getActionSound()**

```
public HSound getActionSound()
```

Associate a sound to be played when the interaction state of the `HGraphicButton` makes the following transitions:

- `NORMAL_STATE` to `ACTIONED_STATE`
- `FOCUSED_STATE` to `ACTIONED_FOCUSED_STATE`

**Specified By:**

`getActionSound()` in interface `HActionable`

**Returns:**

The sound played when the component is actioned.

---

**getDefaultLook()**

```
public static HGraphicLook getDefaultLook()
```

Returns the currently set default `HLook` for `HGraphicButton` components.

**Returns:**

The `HLook` that is used by default when creating a new `HGraphicButton`.

---

**getGainFocusSound()**

```
public HSound getGainFocusSound()
```

Get the sound associated with the gain focus event.

**Specified By:**

`getGainFocusSound()` in interface `HNavigable`

---

**Overrides:**

`getGainFocusSound()` in class `HIcon`

**Returns:**

The sound played when the component gains focus. If no sound is associated with gaining focus, then null shall be returned.

---

**getLoseFocusSound()**

```
public HSound getLoseFocusSound()
```

Get the sound associated with the lost focus event.

**Specified By:**

`getLoseFocusSound()` in interface `HNavigable`

**Overrides:**

`getLoseFocusSound()` in class `HIcon`

**Returns:**

The sound played when the component loses focus. If no sound is associated with losing focus, then null shall be returned.

---

**getMove(int)**

```
public HNavigable getMove(int keyCode)
```

Provides the `HNavigable` object that is navigated to when a particular key is pressed.

**Specified By:**

`getMove(int)` in interface `HNavigable`

**Overrides:**

`getMove(int)` in class `HIcon`

**Parameters:**

`keyCode` - The key code of the pressed key.

**Returns:**

Returns the `HNavigable` object, or if no `HNavigable` is associated with the `keyCode` then returns null.

---

**getNavigationKeys()**

```
public int[] getNavigationKeys()
```

Retrieve the set of key codes which this component maps to navigation targets.

**Specified By:**

`getNavigationKeys()` in interface `HNavigationInputPreferred`

**Overrides:**

`getNavigationKeys()` in class `HIcon`

**Returns:**

an array of key codes, or null if no navigation targets are set on this component.

---

**isSelected()**

```
public boolean isSelected()
```

Indicates if this component has focus.

**Specified By:**

`isSelected()` in interface `HNavigable`

**Overrides:**

`isSelected()` in class `HIcon`

**Returns:**

`true` if the component has focus, otherwise returns `false`.

---

**processHActionEvent(HActionEvent)**

```
public void processHActionEvent(HActionEvent evt)
```

Process an `HActionEvent` sent to this `HGraphicButton`.

**Specified By:**

`processHActionEvent(HActionEvent)` in interface `HActionInputPreferred`

**Parameters:**

`evt` - the `HActionEvent` to process.

---

**processHFocusEvent(HFocusEvent)**

```
public void processHFocusEvent(HFocusEvent evt)
```

Process an `HFocusEvent` sent to this `HGraphicButton`.

**Specified By:**

`processHFocusEvent(HFocusEvent)` in interface `HNavigationInputPreferred`

**Overrides:**

`processHFocusEvent(HFocusEvent)` in class `HIcon`

**Parameters:**

`evt` - the `HFocusEvent` to process.

---

**removeHActionListener(HActionListener)**

```
public void removeHActionListener(HActionListener l)
```

Removes the specified `HActionListener` so that it no longer receives `HActionEvent` events from this `HGraphicButton`. If the specified listener is not registered, the method has no effect. If multiple references to a single listener have been registered it should be noted that this method will only remove one reference per call.

**Specified By:**

`removeHActionListener(HActionListener)` in interface `HActionable`

**Parameters:**

`l` - the `HActionListener`.

---

**setActionCommand(String)**

```
public void setActionCommand(java.lang.String command)
```

Sets the command name for the `HActionEvent` event fired by this `HGraphicButton`.

**Specified By:**

`setActionCommand(String)` in interface `HActionable`

**Parameters:**

`command` - a `String` used to set the action command.

**See Also:**

`getActionCommand()`



---

**setActionSound(HSound)**

```
public void setActionSound(HSound sound)
```

Associate a sound to be played when the interaction state of the `HGraphicButton` makes the following transitions:

- `NORMAL_STATE` to `ACTIONED_STATE`
- `FOCUSED_STATE` to `ACTIONED_FOCUSED_STATE`

**Specified By:**

`setActionSound(HSound)` in interface `HActionable`

**Parameters:**

`sound` - the sound to be played, when the component is actioned. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

---

**setDefaultLook(HGraphicLook)**

```
public static void setDefaultLook(HGraphicLook hlook)
```

Sets the default `HLook` for further `HGraphicButton` components.

**Parameters:**

`hlook` - The `HLook` that will be used by default when creating a new `HGraphicButton` component. Note that this parameter may be null, in which case newly created components shall not draw themselves until a non-null look is set using the `setLook(HLook)` method.

---

**setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)**

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
                             HNavigable right)
```

Set the focus control for an `HNavigable` component. Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is a convenience function for application programmers where a standard up, down, left and right focus traversal between components is required.

Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

Note that this API does not prevent the creation of "isolated" `HNavigable` components --- authors should endeavor to avoid confusing the user.

**Specified By:**

`setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` in interface `HNavigable`

**Overrides:**

`setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` in class `HIcon`

**Parameters:**

`up` - The `HNavigable` component to move to, when the user generates a `VK_UP` `KeyEvent`. If there is no `HNavigable` component to move "up" to, then null should be specified.

`down` - The `HNavigable` component to move to, when the user generates a `VK_DOWN` `KeyEvent`. If there is no `HNavigable` component to move "down" to, then null should be specified.

`left` - The `HNavigable` component to move to, when the user generates a `VK_LEFT` `KeyEvent`. If there is no `HNavigable` component to move "left" to, then null should be specified.

`right` - The `HNavigable` component to move to, when the user generates a `VK_RIGHT` `KeyEvent`. If there is no `HNavigable` component to move "right" to, then null should be specified.

### **setGainFocusSound(HSound)**

```
public void setGainFocusSound(HSound sound)
```

Associate a sound with gaining focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_GAINED`. This sound will start to be played when an object implementing this interface gains focus. It is not guaranteed to be played to completion. If the object implementing this interface loses focus before the audio completes playing, the audio will be truncated. Applications wishing to ensure the audio is always played to completion must implement special logic to slow down the focus transitions.

By default, an `HNavigable` object does not have any gain focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

#### **Specified By:**

`setGainFocusSound(HSound)` in interface `HNavigable`

#### **Overrides:**

`setGainFocusSound(HSound)` in class `HIcon`

#### **Parameters:**

`sound` - the sound to be played, when the component gains focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

### **setLoseFocusSound(HSound)**

```
public void setLoseFocusSound(HSound sound)
```

Associate a sound with losing focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_LOST`. This sound will start to be played when an object implementing this interface loses focus. It is not guaranteed to be played to completion. It is implementation dependent whether and when this sound will be truncated by any gain focus sound played by the next object to gain focus.

By default, an `HNavigable` object does not have any lose focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

#### **Specified By:**

`setLoseFocusSound(HSound)` in interface `HNavigable`

#### **Overrides:**

`setLoseFocusSound(HSound)` in class `HIcon`

#### **Parameters:**

`sound` - the sound to be played, when the component loses focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

### **setMove(int, HNavigable)**

```
public void setMove(int keyCode, HNavigable target)
```

Defines the navigation path from the current `HNavigable` to another `HNavigable` when a particular key is pressed.

Note that `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

**Specified By:**

`setMove(int, HNavigable)` in interface `HNavigable`

**Overrides:**

`setMove(int, HNavigable)` in class `HIcon`

**Parameters:**

`keyCode` - The key code of the pressed key. Any numerical keycode is allowed, but the platform may not be able to generate all keycodes. Application authors should only use keys for which `HRcCapabilities.isSupported()` or `HKeyCapabilities.isSupported()` returns true.

`target` - The target `HNavigable` object that should be navigated to. If a target is to be removed from a particular navigation path, then `null` should be specified.

# org.havi.ui

# HGraphicLook

## Syntax

public class HGraphicLook implements HLook

```
java.lang.Object
|
+--org.havi.ui.HGraphicLook
```

## All Implemented Interfaces:

java.lang.Cloneable, HLook

## Description

The `HGraphicLook` class is used by any `HVisible` component to display static graphical content. This look will be provided by the platform and the exact way in which it is rendered will be platform dependant.

The `HGraphicLook` class draws the content set on an `HVisible`. It uses the `getGraphicContent(int)` method with the current state of the `HVisible` to determine the content to render.

Implementations of `HGraphicLook` should use the appropriate methods on `HVisible` to determine which scaling and alignment modes to use when rendering content. See the class description for `HLook` for more details. `HGraphicLook` may support scalable graphical content. As a minimum, all implementations must support the `RESIZE_NONE` scaling mode, and all alignment modes.

Note that the results of applying the `VALIGN_JUSTIFY` and `HALIGN_JUSTIFY` alignment modes for graphical content are defined to identical to `VALIGN_CENTER` and `HALIGN_CENTER` modes respectively, as justification is meaningless in this context.

This is the default look that is used by `HStaticIcon` and its subclasses. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## See Also:

`HStaticIcon`, `HVisible`, `HLook`

## Constructors

---

## HGraphicLook()

```
public HGraphicLook()
```

Creates a `HGraphicLook` object. See the class description for details of constructor parameters and default values.

## Methods

---

### getInsets(HVisible)

```
public java.awt.Insets getInsets(HVisible visible)
```

Determines the insets of this `HLook`, which indicate the size of the border. This area is reserved for the `HLook` to use for drawing borders around the associated `HVisible`.

**Specified By:**

`getInsets(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

the insets of this `HLook`.

---

### getMaximumSize(HVisible)

```
public java.awt.Dimension getMaximumSize(HVisible hvisible)
```

Gets the maximum size of the `HVisible` component when drawn with this `HLook`.

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook`) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If the `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) then the return value is the current size of the `HVisible` (as returned by `HVisible#getSize`).
2. If the `HLook` does not support scaling of content or no scaling is requested, and content is set then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If there is no content set then a maximum size of `[ Short.MAX_VALUE, Short.MAX_VALUE ]` is returned as a `Dimension`.

**Specified By:**

`getMaximumSize(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating this `HLook` maximum size.

**See Also:**

`getMaximumSize()`

---

**getMinimumSize(HVisible)**

```
public java.awt.Dimension getMinimumSize(HVisible hvisible)
```

Gets the minimum size of the `HVisible` component when drawn with this `HLook`.

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook`) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If the `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) and content is set then the return value is the size of the smallest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
2. If the `HLook` does not support scaling of content or no scaling is requested, *and* content is set then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If no content is available but a default preferred size has been set using `setDefaultSize(Dimension)` has been called to set then the return value is this value (as obtained with `getDefaultSize()`) plus any additional dimensions that the `HLook` requires for border decoration etc.
4. If there is no content or default size set then the return value is an implementation-specific minimum size plus any additional dimensions that the `HLook` requires for border decoration etc.

**Specified By:**

`getMinimumSize(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating this `HLook` minimum size.

**See Also:**

`getMinimumSize()`

---

**getPreferredSize(HVisible)**

```
public java.awt.Dimension getPreferredSize(HVisible hvisible)
```

Gets the preferred size of the `HVisible` component when drawn with this `HLook`.

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook`) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If a default preferred size has been set for this `HVisible` (using `setDefaultSize(Dimension)`) then the return value is this size (obtained with `getDefaultSize()`) plus any additional dimensions that the `HLook` requires for border decoration etc.
2. If this `HLook` does not support scaling of content or no scaling is requested, and content is present then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If this `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) and content is set then the

return value is the current size of the `HVisible` as returned by `getSize` ).

- If there is no content and no default size set then the return value is the current size of the `HVisible` as returned by `getSize` ).

**Specified By:**

`getPreferredSize(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating the preferred size of the `HVisible` when drawn with this `HLook` .

**See Also:**

`getPreferredSize()` , `setDefaultSize(Dimension)`

### **isOpaque(HVisible)**

```
public boolean isOpaque(HVisible visible)
```

Returns true if the entire painted area of the `HVisible` when using this look is fully opaque, i.e. the `showLook(Graphics, HVisible, int)` method guarantees that all pixels are painted in an opaque `Color`.

The default value is implementation specific and depends on the background painting mode of the given `HVisible` . The consequences of an invalid overridden value are implementation specific.

**Specified By:**

`isOpaque(HVisible)` in interface `HLook`

**Parameters:**

`visible` - the visible to test

**Returns:**

true if all the pixels with the `java.awt.Component#getBounds` method of an `HVisible` using this look are fully opaque, i.e. the `showLook(Graphics, HVisible, int)` method guarantees that all pixels are painted in an opaque `Color`, otherwise false.

### **showLook(Graphics, HVisible, int)**

```
public void showLook(java.awt.Graphics g, HVisible visible, int state)
```

The `showLook(Graphics, HVisible, int)` method is responsible for repainting the entire `HVisible` component, (including any content set on the component, and the component background), subject to the `clipRect` of the `Graphics` object passed to it.

The `showLook(Graphics, HVisible, int)` method should not modify the `clipRect` of the `Graphics` object that is passed to it.

For looks which draw content (e.g. `HTextLook` , `HGraphicLook` and `HAnimateLook` ), if no content is associated with the component, the `showLook(Graphics, HVisible, int)` method paints the component with its current background `Color` according to the `setBackgroundMode(int)` method of `HVisible` and draws any (implementation-specific) borders. Note that by default the background mode is set so as to **not** paint a background. Furthermore on platforms which support transparent colors the background `Color` may be partially or completely transparent.

Any resources **explicitly** associated with an `HLook` should be loaded by the `HLook` during its creation, etc., or via its `setXXX()` methods. Note that the "standard" looks don't load content by default.

This method is called from the `paint(Graphics)` method of `HVisible` and must never be called from elsewhere. Components wishing to redraw themselves should call their repaint method in the usual way.

**Specified By:**

```
showLook(Graphics, HVisible, int) in interface HLook
```

**Parameters:**

`g` - the graphics context.

`visible` - the visible.

`state` - the state parameter indicates the state of the visible, allowing the look to render the appropriate content for that state. Note that some components (e.g. `HStaticRange`, `HRange`, `HRangeValue`) do not use state-based content).

---

**widgetChanged(HVisible, HChangeData[])**

```
public void widgetChanged(HVisible visible, HChangeData[] changes)
```

Called by the `HVisible` whenever its content, state, or any other data changes. See the class description of `HVisible` for more information about the `changes` parameter.

The implementation of this method should work out which graphical areas of the `HVisible` have changed and make any relevant calls to trigger the repainting of those areas.

A minimum implementation of this method could simply call

```
visible.repaint()
```

**Specified By:**

```
widgetChanged(HVisible, HChangeData[] ) in interface HLook
```

**Parameters:**

`visible` - the `HVisible` which has changed

`changes` - an array containing hint data and associated hint objects. If this argument is `null` a full repaint will be triggered.



# org.havi.ui HGraphicsConfigTemplate

## Syntax

```
public class HGraphicsConfigTemplate extends HScreenConfigTemplate
```

```
java.lang.Object
|
+--HScreenConfigTemplate
|
+--org.havi.ui.HGraphicsConfigTemplate
```

## Description

The `HGraphicsConfigTemplate` class is used to obtain a valid `HGraphicsConfiguration`. An application instantiates one of these objects and then sets all non-default attributes as desired. The `getBestConfiguration(HGraphicsConfigTemplate)` method found in the `HGraphicsDevice` class is then called with this `HGraphicsConfigTemplate`. A valid `HGraphicsConfiguration` is returned that meets or exceeds what was requested in the `HGraphicsConfigTemplate`.

This class may be subclassed to support additional properties of graphics configurations which may be requested by applications. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## See Also:

`HScreenConfigTemplate`, `HBackgroundConfigTemplate`, `HVideoConfigTemplate`

## Fields

### IMAGE\_SCALING\_SUPPORT

```
public static final int IMAGE_SCALING_SUPPORT
```

A value for use in the preference field of the `setPreference(int, int)` and `getPreferencePriority(int)` methods in the `HGraphicsConfigTemplate` that indicates that the graphics configuration should or shall support rapid (hardware) image scaling.

### MATTE\_SUPPORT

```
public static final int MATTE_SUPPORT
```

A value for use in the preference field of the `setPreference(int, int)` and `getPreferencePriority(int)` methods in the `HGraphicsConfigTemplate` that indicates that the graphics configuration should or shall support the HAVi mattes feature.

---

## VIDEO\_MIXING

```
public static final int VIDEO_MIXING
```

A value for use in the preference field of the `setPreference(int, int)` and `getPreferencePriority(int)` methods in the `HGraphicsConfigTemplate` that indicates that the graphics configuration should or shall support transparency in the graphics system such that the output of a video decoder is visible. This includes the following configurations :-

- Configurations where there is a well defined transformation between video pixels and graphics pixels (e.g. pixels are the same size).
- Configurations where an application displays graphics over video but where the video is considered as a background and hence no transformation between the two sets of pixels is required.

Applications may specify a particular video configuration with which mixing must be supported. In this case, the video configuration is specified as an `HVideoConfiguration` object. If no specific video configuration is required then it is not required to specify such a configuration and null can be used.

This preference is used by the platform as a constraint in selecting configurations. Templates generated by the platform and then returned to applications (e.g. from a `getConfigTemplate` method) shall not have this preference filled in by the platform.

## Constructors

---

### HGraphicsConfigTemplate()

```
public HGraphicsConfigTemplate()
```

Creates an `HGraphicsConfigTemplate` object. See the class description for details of constructor parameters and default values.

## Methods

---

### getPreferenceObject(int)

```
public java.lang.Object getPreferenceObject(int preference)
```

Return the preference object for the specified preference.

Instances of `HGraphicsConfigTemplate` which have not had this preference set shall return null for this object. Note that instances constructed by the platform and returned to applications are required to have all preferences (except where explicitly identified) set by the platform before it is returned.

#### Overrides:

`getPreferenceObject(int)` in class `HScreenConfigTemplate`

**Parameters:**

preference - the preference to be indicated. Valid values for an `HGraphicsConfigTemplate` are: `VIDEO_GRAPHICS_PIXEL_ALIGNED`, `PIXEL_ASPECT_RATIO`, `PIXEL_RESOLUTION`, `SCREEN_RECTANGLE`, and `VIDEO_MIXING`.

Subclasses may add further valid values. A `IllegalArgumentException` shall be thrown if the preference is not a valid value for this instance of `HGraphicsConfigTemplate`, or if preference does not have an associated value object.

**Returns:**

the preference object for the specified preference.

**getPreferencePriority(int)**

```
public int getPreferencePriority(int preference)
```

Return the priority for the specified preference.

By default the preferences in a template returned from the system will have an `DONT_CARE` priority unless specified otherwise. Any configuration always satisfies these attributes.

**Overrides:**

`getPreferencePriority(int)` in class `HScreenConfigTemplate`

**Parameters:**

preference - the preference to be indicated. Valid values for an `HGraphicsConfigTemplate` are: `ZERO_BACKGROUND_IMPACT`, `ZERO_GRAPHICS_IMPACT`, `ZERO_VIDEO_IMPACT`, `INTERLACED_DISPLAY`, `FLICKER_FILTERING`, `VIDEO_GRAPHICS_PIXEL_ALIGNED`, `PIXEL_ASPECT_RATIO`, `PIXEL_RESOLUTION`, `SCREEN_RECTANGLE`, `VIDEO_MIXING`, `MATTE_SUPPORT` and `IMAGE_SCALING_SUPPORT`.

Subclasses may add further valid values. An `IllegalArgumentException` shall be thrown if the preference is not a valid value for this instance of `HGraphicsConfigTemplate`.

**Returns:**

the priority for the specified preference.

**isConfigSupported(HGraphicsConfiguration)**

```
public boolean isConfigSupported(HGraphicsConfiguration hgc)
```

Returns a boolean indicating whether or not the specified `HGraphicsConfiguration` can be used to create a graphics plane that supports the features set in this template.

**Parameters:**

hgc -- the `HGraphicsConfiguration` object to test against this template.

**Returns:**

true if this `HGraphicsConfiguration` object can be used to create a graphics plane that supports the features set in this template, false otherwise.

**setPreference(int, int)**

```
public void setPreference(int preference, int priority)
```

Set the indicated preference to have the specified priority. If the preference has been previously set, then the previous priority for the preference shall be overwritten.

Attributes that are not filled in in a template (through `setPreference(int, int)`), shall have the priority `DONT_CARE`. Any configuration always satisfies these attributes.

**Overrides:**

`setPreference(int, int)` in class `HScreenConfigTemplate`

**Parameters:**

`preference` - the preference to be indicated. Valid values for an `HGraphicsConfigTemplate` are: `ZERO_BACKGROUND_IMPACT`, `ZERO_GRAPHICS_IMPACT`, `ZERO_VIDEO_IMPACT`, `INTERLACED_DISPLAY`, `FLICKER_FILTERING`, `MATTE_SUPPORT` and `IMAGE_SCALING_SUPPORT`.

Subclasses may add further valid values. An `IllegalArgumentException` shall be thrown if the preference is not a valid value for this instance of `HGraphicsConfigTemplate`

`priority` - the priority of the preference. Valid values include: `REQUIRED`, `PREFERRED`, `DONT_CARE`, `PREFERRED_NOT` and `REQUIRED_NOT`.

If `priority` is not a valid priority as defined here a `java.lang.IllegalArgumentException` will be thrown.

**setPreference(int, Object, int)**

`public void setPreference(int preference, java.lang.Object object, int priority)`

Set the indicated preference (and associated value object) to have the specified priority. If the preference has been previously set, then the previous object and priority shall be overwritten.

Attributes that are not filled in in a template (through `setPreference(int, int)`), shall have the priority `DONT_CARE`. Any configuration always satisfies these attributes.

An application which wishes to remove a preference from an existing template (e.g. one generated by the platform) may call this method with null for the object parameter.

**Overrides:**

`setPreference(int, Object, int)` in class `HScreenConfigTemplate`

**Parameters:**

`preference` - the preference to be indicated. Valid values for an `HScreenConfigTemplate` are: `PIXEL_ASPECT_RATIO`, `PIXEL_RESOLUTION`, `SCREEN_RECTANGLE`, `VIDEO_GRAPHICS_PIXEL_ALIGNED` and `VIDEO_MIXING`.

Subclasses may add further valid values. An `IllegalArgumentException` shall be thrown if the preference is not a valid value for this instance of `HScreenConfigTemplate`

`object` - the Object associated with the given preference, or null.

`priority` - the priority of the preference. Valid values include: `REQUIRED`, `PREFERRED`, `DONT_CARE`, `PREFERRED_NOT` and `REQUIRED_NOT`.

If `priority` is not a valid priority as defined here a `java.lang.IllegalArgumentException` will be thrown.

# org.havi.ui HGraphicsConfiguration

## Syntax

```
public class HGraphicsConfiguration extends HScreenConfiguration
```

```
java.lang.Object
|
+--HScreenConfiguration
|
+--org.havi.ui.HGraphicsConfiguration
```

## Direct Known Subclasses:

```
HEmulatedGraphicsConfiguration
```

## Description

The `HGraphicsConfiguration` class describes the characteristics (settings) of an `HGraphicsDevice`. There can be many `HGraphicsConfiguration` objects associated with a single `HGraphicsDevice`. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
-----------	-------------	---------------	------------	------------

None.

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

None.

## See Also:

```
HGraphicsDevice
```

## Constructors

### HGraphicsConfiguration()

```
protected HGraphicsConfiguration()
```

It is not intended that applications should directly construct `HGraphicsConfiguration` objects.

Creates an `HGraphicsConfiguration` object. See the class description for details of constructor parameters and default values.

## Methods

---

**dispose(Color)**

```
public void dispose(java.awt.Color c)
```

This method is used by an application when a color returned from those versions of the method `getPunchThroughToBackgroundColor` with a `Color` as a parameter is no longer required. It is the responsibility of applications to ensure that no pixels which they had drawn using this color are still displayed on the screen before calling this method. The result of using such a `Color` after calling this method is implementation dependent. Using a color obtained from another source apart from the specified methods will result in this method having no effect.

**Parameters:**

`c` - the `Color` which is no longer required.

---

**getAllFonts()**

```
public java.awt.Font[] getAllFonts()
```

List the fonts that are always available on the device, but does not list fonts that may be (temporarily) available for download from other sources.

**Returns:**

an array of `java.awt.Font` objects which are always available on the device.

---

**getCompatibleImage(Image, HImageHints)**

```
public java.awt.Image getCompatibleImage(java.awt.Image input, HImageHints ih)
```

Generate a `java.awt.Image` which *may* be a modified copy of the image passed as `input`. Such a copy is modified as necessary such that it is compatible with the current `HGraphicsConfiguration`. For example this may involve dithering the image to a restricted color palette. In the case where no modification is required a reference to the original image `input` will be returned instead of a separate new `java.awt.Image`.

Note: Unmodified Images, or Images modified for other `HGraphicsConfiguration` should still be able to be rendered within this `HGraphicsConfiguration`, but may not be as efficient (rapid) in terms of rendering, and may not be presented optimally. For example, an 8 bit per RGB pixel image loaded onto a configuration with a 4 bit per RGB pixel framebuffer may have its pixel values truncated, if this Image is then displayed on an alternate configuration with 16 bits per RGB pixel then it will obviously not be displayed optimally.

The `HImageHints` provide a mechanism to indicate how any conversion to a constrained graphics environment might best be performed, by describing the general image contents.

It is implementation (and algorithmically) dependent whether this method operates on partial, or complete Image pixel data.

**Parameters:**

`input` - the `java.awt.Image` to be modified

`ih` - an `HImageHints` object that indicates the expected type of the input Image, so that its presentation can be optimally adjusted.

**Returns:**

a `java.awt.Image` which has been determined to be optimally suited for presentation on the `HGraphicsDevice` associated with this `HGraphicsConfiguration`. Note that on some `HGraphicsConfiguration` a reference to the original Image may be returned, this is especially true for systems with high-end graphics capabilities.

---

**getComponentHScreenRectangle(Component)**

```
public HScreenRectangle getComponentHScreenRectangle(java.awt.Component component)
```

Returns the on-screen location of a given visible `java.awt.Component` as an `HScreenRectangle` for this `HGraphicsDevice`.

**Parameters:**

`component` - the `java.awt.Component` whose on-screen area is to be determined.

**Returns:**

the on-screen location of component as an `HScreenRectangle` for this `HGraphicsDevice`, or null if the component is not currently added to the `HScene` (or one of its "child" containers).

**See Also:**

`HScreenRectangle`

---

**getConfigTemplate()**

```
public HGraphicsConfigTemplate getConfigTemplate()
```

Returns an `HGraphicsConfigTemplate` object that describes and uniquely identifies this `HGraphicsConfiguration`.

Hence, the following sequence should return the original `HGraphicsConfiguration`.

```
HGraphicsDevice.getBestMatch(HGraphicsConfiguration.getConfigTemplate())
```

Features that are implemented in the `HGraphicsConfiguration` will return `REQUIRED` priority. Features that are not implemented in the `HGraphicsConfiguration` will return `REQUIRED_NOT` priority.

**Returns:**

an `HGraphicsConfigTemplate` object which both describes and uniquely identifies this `HGraphicsConfiguration`.

---

**getDevice()**

```
public HGraphicsDevice getDevice()
```

Returns the `HGraphicsDevice` associated with this `HGraphicsConfiguration`.

**Returns:**

the `HGraphicsDevice` object that is associated with this `HGraphicsConfiguration`,

---

**getPixelCoordinatesHScreenRectangle(HScreenRectangle, Container)**

```
public java.awt.Rectangle getPixelCoordinatesHScreenRectangle(HScreenRectangle sr,
    java.awt.Container cont)
```

Returns a `java.awt.Rectangle` which contains the graphics (AWT) pixel area for an `HScreenRectangle` relative to the supplied `java.awt.Container`.

**Parameters:**

`sr` - the screen location expressed as an `HScreenRectangle`.

`cont` - the `java.awt.Container` in whose coordinate system the screen location should be expressed.

**Returns:**

a `java.awt.Rectangle` which contains the graphics (AWT) pixel area for an `HScreenRectangle` relative to the supplied `java.awt.Container`. The returned x, y, width, height values in the `java.awt.Rectangle` should be such that a

- `r = getPixelCoordinatesHScreenRectangle(sr, cont);`
- `cont.add(component);`
- `component.setBounds(r.x, r.y, r.width, r.height);`  
should ensure that the dimensions of the component on-screen should correspond to the given `HScreenRectangle` , subject to clipping by its parent container, `cont`.

Note that the `HScreenRectangle` (`HScreenPoint`) coordinates are in floats - conversion to pixel coordinate systems necessarily implies a potential loss of precision - however, such conversion should be to the "nearest" integer pixel coordinate.

### **getPunchThroughToBackgroundColor(Color, int)**

```
public java.awt.Color getPunchThroughToBackgroundColor(java.awt.Color color,
    int percentage)
```

This method returns a `Color` that may be used in standard graphics drawing operations, which has the effect of "punching through" all Components that are behind the Component in which the drawing operation is performed. This includes any visual Components acquired from JMF players. What is behind this `HGraphicsConfiguration` is revealed through the drawn "hole" blended with the graphics color specified as the first parameter to this method. Platforms with restricted color spaces may make approximations as required to obtain the best match possible.

#### **Parameters:**

`color` - the graphics color to blend

`percentage` - the blending value for this color with respect to what is outside this `HGraphicsConfiguration` . The specified value will be clamped to the range 0 to 100.

#### **Returns:**

a `Color` with the desired effect or null for configurations which do not or are currently unable to support this rendering mode.

### **getPunchThroughToBackgroundColor(Color, int, HVideoDevice)**

```
public java.awt.Color getPunchThroughToBackgroundColor(java.awt.Color color,
    int percentage, HVideoDevice v)
```

This method returns a `Color` that may be used in standard graphics drawing operations, which has the effect of modifying the existing color of a pixel to make it partially (or wholly) transparent to the background. The existing pixel percentage transparency to the background at that point shall be equivalent to the (closest) percentage value as specified in the `getPunchThroughToBackgroundColor` percentage parameter.

The existing RGB values of the pixel are unchanged as far as possible, within the limits of the platform. Platforms with restricted color spaces may make approximations as required to obtain the best possible match.

The precise contents of the background are as defined by the platform including any `HBackgroundDevice` , etc.

#### **Parameters:**

`color` - the graphics color to blend

`percentage` - the alpha value for this color with respect to what is outside this `HGraphicsConfiguration` . The specified value will be clamped to the range 0 to 100.

#### **Returns:**

a `Color` with the desired effect or null for configurations which do not or are currently unable to support this rendering mode.



---

**getPunchThroughToBackgroundColor(int)**

```
public java.awt.Color getPunchThroughToBackgroundColor(int percentage)
```

This method returns a `Color` that may be used in standard graphics drawing operations, which has the effect of modifying the existing color of a pixel to make it partially (or wholly) transparent to the background. The existing pixel percentage transparency to the background at that point shall be equivalent to the (closest) percentage value as specified in the `getPunchThroughToBackgroundColor` percentage parameter. A value of 0% is fully transparent and 100% is fully opaque.

The existing RGB values of the pixel are unchanged as far as possible, within the limits of the platform. Platforms with restricted color spaces may make approximations as required to obtain the best possible match.

The precise contents of the background are as defined by the platform including any `HBackgroundDevice`, etc.

**Parameters:**

`percentage` - the new blending value for each pixel drawn with this color with respect to what is outside this `HGraphicsConfiguration`. The specified value will be clamped to the range 0 to 100.

**Returns:**

a `Color` with the desired effect or null for configurations which do not or are currently unable to support this rendering mode.

---

**getPunchThroughToBackgroundColor(int, HVideoDevice)**

```
public java.awt.Color getPunchThroughToBackgroundColor(int percentage, HVideoDevice hvd)
```

This method returns a `Color` that may be used in standard graphics drawing operations, which has the effect of "punching through" the `HGraphicsDevice` in which the drawing operation is performed. The specified `HVideoDevice` is revealed through the drawn "hole". The value specified replaces the blending value (with respect to this `HVideoDevice`) of each pixel drawn with this color. The existing RGB values of the pixel are unchanged as far as possible within the limits of the platform. Platforms with restricted color spaces may make approximations as required to obtain the best match possible.

**Parameters:**

`percentage` - the new alpha value for each pixel drawn with this color with respect to the the `HVideoDevice` specified. The specified value will be clamped to the range 0 to 100.

`hvd` - the `HVideoDevice` to reveal.

**Returns:**

a `Color` with the desired effect or null for configurations which do not or are currently unable to support this rendering mode.

# org.havi.ui HGraphicsDevice

## Syntax

```
public class HGraphicsDevice extends HScreenDevice
```

```
java.lang.Object
|
+--HScreenDevice
|
+--org.havi.ui.HGraphicsDevice
```

## Direct Known Subclasses:

```
HEmulatedGraphicsDevice
```

## All Implemented Interfaces:

```
org.davic.resources.ResourceProxy, org.davic.resources.ResourceServer
```

## Description

The `HGraphicsDevice` class describes the raster graphics devices that are available for a particular `HScreen`. Each `HGraphicsDevice` has one or more `HGraphicsConfiguration` objects associated with it. These objects specify the different configurations (settings) in which the `HGraphicsDevice` can be used. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## See Also:

```
HGraphicsConfiguration, HScreenDevice, HScreen
```

## Constructors

### HGraphicsDevice()

```
protected HGraphicsDevice()
```

It is not intended that applications should directly construct `HGraphicsDevice` objects.

Creates an `HGraphicsDevice` object. See the class description for details of constructor parameters and default values.

## Methods

---

### getBestConfiguration(HGraphicsConfigTemplate)

```
public HGraphicsConfiguration getBestConfiguration(HGraphicsConfigTemplate hgct)
```

The `getBestConfiguration` method attempts to return an `HGraphicsConfiguration` that matches the specified `HGraphicsConfigTemplate`. If this is not possible it will attempt to construct an `HEmulatedGraphicsConfiguration` where the emulated configuration best matches this `HGraphicsConfigTemplate`. If this is not possible, null is returned. Note that the algorithm used for this behavior, and the extent and behavior of `HEmulatedGraphicsConfiguration` are platform specific.

Equally best in this sense means that the configurations satisfy an equal number of preferences with priorities `PREFERRED` and `PREFERRED_NOT` and all preferences with priorities `REQUIRED` and `REQUIRED_NOT`.

Configurations are chosen according to the following algorithm, based on the priority as supplied to `setPreference(int, int)`. Configurations must:

1. satisfy ALL the preferences whose priority was `REQUIRED`
2. satisfy NONE of the preferences whose priority was `REQUIRED_NOT`
3. satisfy as many as possible of the preferences whose priority was `PREFERRED`.
4. satisfy as few as possible of the preferences whose priority was `PREFERRED_NOT`.

Preferences whose priority was `DONT_CARE` are ignored.

This method returns null if no configuration exists that satisfies all `REQUIRED` and `REQUIRED_NOT` priorities.

#### Parameters:

`hgct` - an `HGraphicsConfigTemplate` object used to obtain a valid `HGraphicsConfiguration`. If this parameter is null the default configuration for the platform shall be returned.

#### Returns:

an `HGraphicsConfiguration` that passes the criteria defined in the specified `HGraphicsConfigTemplate` or null if no `HGraphicsConfiguration` or `HEmulatedGraphicsConfiguration` passes the criteria.

---

### getBestConfiguration(HGraphicsConfigTemplate[])

```
public HGraphicsConfiguration getBestConfiguration(HGraphicsConfigTemplate[] hgcta)
```

The `getBestConfiguration` method attempts to return an `HGraphicsConfiguration` that matches the specified `HGraphicsConfigTemplate` objects within the specified array. The `HGraphicsTemplate` objects should be considered for matching in priority order from 0 to  $(hgcta.length - 1)$ . If this is not possible, it will attempt to construct an `HEmulatedGraphicsConfiguration` where the emulated configuration best matches the specified `HGraphicsConfigTemplate` objects. If this is not possible, null is returned.

Equally best in this sense means that the configurations satisfy an equal number of preferences with priorities `PREFERRED` and `PREFERRED_NOT` and all preferences with priorities `REQUIRED` and `REQUIRED_NOT`.

Configurations are chosen according to the following algorithm, based on the priority as supplied to `setPreference(int, int)`. Configurations must:

1. satisfy ALL the preferences whose priority was `REQUIRED`
2. satisfy NONE of the preferences whose priority was `REQUIRED_NOT`

3. satisfy as many as possible of the preferences whose priority was `PREFERRED` .
4. satisfy as few as possible of the preferences whose priority was `PREFERRED_NOT` .

Preferences whose priority was `DONT_CARE` are ignored.

This method returns null if no configuration exists that satisfies all `REQUIRED` and `REQUIRED_NOT` priorities.

**Parameters:**

`hgcta` - the `HGraphicsConfigTemplate` array used to obtain a valid `HGraphicsConfiguration` .

**Returns:**

an `HGraphicsConfiguration` that passes the criteria defined in one of the `HGraphicsConfigTemplate` objects within the specified array. The class of the object returned can also be a class inheriting from `HGraphicsConfiguration` (e.g. `HEmulatedGraphicsConfiguration` ).

---

### **getConfigurations()**

```
public HGraphicsConfiguration[] getConfigurations()
```

Returns all of the `HGraphicsConfiguration` objects associated with this `HGraphicsDevice` .

**Returns:**

an array of `HGraphicsConfiguration` objects that are associated with this `HGraphicsDevice` . The class of the objects returned can also be a class inheriting from `HGraphicsConfiguration` (e.g. `HEmulatedGraphicsConfiguration` ).

**See Also:**

`HGraphicsConfiguration`

---

### **getCurrentConfiguration()**

```
public HGraphicsConfiguration getCurrentConfiguration()
```

Returns the current `HGraphicsConfiguration` for this `HGraphicsDevice` .

**Returns:**

the current `HGraphicsConfiguration` for this `HGraphicsDevice` . The class of the object returned can also be a class inheriting from `HGraphicsConfiguration` (e.g. `HEmulatedGraphicsConfiguration` ).

**See Also:**

`HGraphicsConfiguration`

---

### **getDefaultConfiguration()**

```
public HGraphicsConfiguration getDefaultConfiguration()
```

Returns the default `HGraphicsConfiguration` associated with this `HGraphicsDevice` . This (single) default configuration should correspond to some well-behaved settings for the device, such as, a minimal configuration, or factory preset settings.

**Returns:**

the default `HGraphicsConfiguration` of this `HGraphicsDevice` . The class of the object returned can also be a class inheriting from `HGraphicsConfiguration` (e.g. `HEmulatedGraphicsConfiguration` ).

## setGraphicsConfiguration(HGraphicsConfiguration)

```
public boolean setGraphicsConfiguration(HGraphicsConfiguration hgc)
```

Set the graphics configuration for the device.

An application is only allowed to call this method after it reserved the device explicitly and subject to the security policy of the platform. Subject to this, the following rules determine whether this method can succeed.

- If the application selects a configuration that is not conflicting with the configurations of all the other devices on the same `HScreen` then that configuration is selected.
- If an application tries to select a configuration that conflicts with a configuration of another device on this `HScreen` which this application cannot control due to platform security policy then this method shall fail with a `SecurityException`.
- If an application tries to select a configuration that conflicts with a configuration of another device on this `HScreen` which this application cannot control due to another application owning the right to control that device and the platform not giving that right to this application then this method shall fail with an `HPermissionDeniedException`.
- If an application tries to select a configuration that conflicts with a configuration of another device which this application can control due to either this or no application having reserved the device, then the configuration of the other device is automatically changed. There is no change in the resource ownership of the other device.

Applications can prevent or limit changes to configurations of other, not intended, devices by using constants `ZERO_GRAPHICS_IMPACT` and `ZERO_VIDEO_IMPACT` in their configuration templates. The first one will cause no changes to already running graphical applications. This means that no changes may be applied to the graphics device. Similarly the second constant will result in no changes to the video device.

Any modifications made to the configurations of other devices shall be reflected by the API for the devices concerned. The `getCurrentConfiguration()` method for those devices shall return the new configuration.

On successful change to the specified configuration, the device shall fire one or more `HScreenConfigurationEvent` for all listeners that are currently registered (if the criteria for receiving such an event has been satisfied). If the new configuration differs by more than one characteristic from the previous configuration then the device may fire one or more `HScreenConfigurationEvent`. This behavior is implementation specific.

If an attempt is made to set the identical configuration to that which is current, then no such event(s) shall be fired.

### Parameters:

`hgc` - the `HGraphicsConfiguration` to which this device should be set.

### Returns:

A boolean indicating whether the configuration could be applied successfully. If the configuration could not be applied successfully, the configuration after this method may not match the configuration of the device prior to this method being called --- applications should take steps to determine whether a partial change of settings has been made.

### Throws:

`SecurityException` - if the application does not have sufficient rights to set the configuration for this device.

{ @link - org.havi.ui.HPermissionDeniedException HPermissionDeniedException} if the application does not currently have the right to set the configuration for this device.

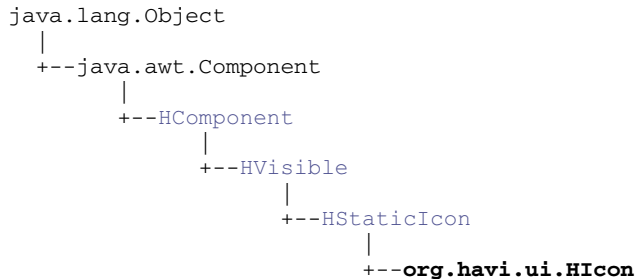
{ @link - org.havi.ui.HConfigurationException HConfigurationException} if the specified configuration is not valid for this device, or if it conflicts with other devices whose configuration(s) cannot be changed by this application.

`HConfigurationException`, `HPermissionDeniedException`

# org.havi.ui HIcon

## Syntax

public class HIcon extends HStaticIcon implements HNavigable



## Direct Known Subclasses:

HGraphicButton

## All Implemented Interfaces:

HMatteLayer, HNavigable, HNavigationInputPreferred, HNoInputPreferred, HState, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, TestOpacity

## Description

The `HIcon` is a user interface component used to display static graphical content (as `HStaticIcon`) which also enables a user to navigate to it, i.e. it can have the input focus. By default it uses the `HGraphicLook` to render itself. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
height	height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
image	The image to be used as the content for every state of this component.	null	setGraphicContent(Image, int)	getGraphicContent(int)

imageNormal	The image to be used as the content for the <code>NORMAL_STATE</code> state of this component.	null	<code>setGraphicContent(Image, int)</code>	<code>getGraphicContent(int)</code>
imageFocus	The image to be used as the content for the focused states of this component.	null	<code>setGraphicContent(Image, int)</code>	<code>getGraphicContent(int)</code>

#### Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte ( <code>HMatte</code> ).	none (i.e. <code>getMatte()</code> returns null)	<code>setMatte(HMatte)</code>	<code>getMatte()</code>
The text layout manager responsible for text formatting.	An <code>HDefaultTextLayoutManager</code> object.	<code>setTextLayoutManager(HTextLayoutManager)</code>	<code>getTextLayoutManager()</code>
The background painting mode	<code>NO_BACKGROUND_FILL</code>	<code>setBackgroundMode(int)</code>	<code>getBackgroundMode()</code>
The default preferred size	not set (i.e. null) unless specified by width and height parameters	<code>setDefaultSize(Dimension)</code>	<code>getDefaultSize()</code>
The horizontal content alignment	<code>HALIGN_CENTER</code>	<code>setHorizontalAlignment(int)</code>	<code>getHorizontalAlignment()</code>
The vertical content alignment	<code>VALIGN_CENTER</code>	<code>setVerticalAlignment(int)</code>	<code>getVerticalAlignment()</code>
The content scaling mode	<code>RESIZE_NONE</code>	<code>setResizeMode(int)</code>	<code>getResizeMode()</code>
The default "look" for this class.	A platform specific <code>HGraphicLook</code>	<code>setDefaultLook(HGraphicLook)</code>	<code>getDefaultLook()</code>
The "look" for this object.	The <code>HGraphicLook</code> returned from <code>HIcon.getDefaultLook</code> when this object was created.	<code>setLook(HLook)</code>	<code>getLook()</code>
The gain focus sound.	null	<code>setGainFocusSound(HSound)</code>	<code>getGainFocusSound()</code>
The lose focus sound.	null	<code>setLoseFocusSound(HSound)</code>	<code>getLoseFocusSound()</code>

#### See Also:

`HStaticIcon`, `HNavigable`

## Constructors

---

### HIcon()

```
public HIcon()
```

Creates an `HIcon` object. See the class description for details of constructor parameters and default values.

---

### HIcon(Image)

```
public HIcon(java.awt.Image image)
```

Creates an `HIcon` object. See the class description for details of constructor parameters and default values.

---

### HIcon(Image, Image, int, int, int, int)

```
public HIcon(java.awt.Image imageNormal, java.awt.Image imageFocus, int x, int y,
             int width, int height)
```

Creates an `HIcon` object. See the class description for details of constructor parameters and default values.

---

### HIcon(Image, int, int, int, int)

```
public HIcon(java.awt.Image image, int x, int y, int width, int height)
```

Creates an `HIcon` object. See the class description for details of constructor parameters and default values.

---

## Methods

---

### getDefaultLook()

```
public static HGraphicLook getDefaultLook()
```

Returns the currently set default look for `HIcon` components.

**Returns:**

The look that is used by default when creating a new `HIcon` component.

---

### getGainFocusSound()

```
public HSound getGainFocusSound()
```

Get the sound associated with the gain focus event.

**Specified By:**

`getGainFocusSound()` in interface `HNavigable`

**Returns:**

The sound played when the component gains focus. If no sound is associated with gaining focus, then null shall be returned.

---



---

**getLoseFocusSound()**

```
public HSound getLoseFocusSound()
```

Get the sound associated with the lost focus event.

**Specified By:**

```
getLoseFocusSound() in interface HNavigable
```

**Returns:**

The sound played when the component loses focus. If no sound is associated with losing focus, then null shall be returned.

---

**getMove(int)**

```
public HNavigable getMove(int keyCode)
```

Provides the `HNavigable` object that is navigated to when a particular key is pressed.

**Specified By:**

```
getMove(int) in interface HNavigable
```

**Parameters:**

`keyCode` - The key code of the pressed key.

**Returns:**

Returns the `HNavigable` object, or if no `HNavigable` is associated with the `keyCode` then returns null.

---

**getNavigationKeys()**

```
public int[] getNavigationKeys()
```

Retrieve the set of key codes which this component maps to navigation targets.

**Specified By:**

```
getNavigationKeys() in interface HNavigationInputPreferred
```

**Returns:**

an array of key codes, or null if no navigation targets are set on this component.

---

**isSelected()**

```
public boolean isSelected()
```

Indicates if this component has focus.

**Specified By:**

```
isSelected() in interface HNavigable
```

**Returns:**

true if the component has focus, otherwise returns false.

---

**processHFocusEvent(HFocusEvent)**

```
public void processHFocusEvent(HFocusEvent evt)
```

Process an `HFocusEvent` sent to this `HIcon`.

**Specified By:**

```
processHFocusEvent(HFocusEvent) in interface HNavigationInputPreferred
```

**Parameters:**

`evt` - the `HFocusEvent` to process.

---

**setDefaultLook(HGraphicLook)**

```
public static void setDefaultLook(HGraphicLook hlook)
```

Sets the default `HLook` for further `HIcon` Components.

**Parameters:**

`hlook` - The `HLook` that will be used by default when creating a new `HIcon` component. Note that this parameter may be null, in which case newly created components shall not draw themselves until a non-null look is set using the `setLook(HLook)` method.

---

**setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)**

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
                             HNavigable right)
```

Set the focus control for an `HNavigable` component. Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is a convenience function for application programmers where a standard up, down, left and right focus traversal between components is required.

Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

Note that this API does not prevent the creation of "isolated" `HNavigable` components --- authors should endeavor to avoid confusing the user.

**Specified By:**

`setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` in interface `HNavigable`

**Parameters:**

`up` - The `HNavigable` component to move to, when the user generates a `VK_UP` KeyEvent. If there is no `HNavigable` component to move "up" to, then null should be specified.

`down` - The `HNavigable` component to move to, when the user generates a `VK_DOWN` KeyEvent. If there is no `HNavigable` component to move "down" to, then null should be specified.

`left` - The `HNavigable` component to move to, when the user generates a `VK_LEFT` KeyEvent. If there is no `HNavigable` component to move "left" to, then null should be specified.

`right` - The `HNavigable` component to move to, when the user generates a `VK_RIGHT` KeyEvent. If there is no `HNavigable` component to move "right" to, then null should be specified.

---

**setGainFocusSound(HSound)**

```
public void setGainFocusSound(HSound sound)
```

Associate a sound with gaining focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_GAINED`. This sound will start to be played when an object implementing this interface gains focus. It is not guaranteed to be played to completion. If the object implementing this interface loses focus before the audio completes playing, the audio will be truncated. Applications wishing to ensure the audio is always played to completion must implement special logic to slow down the focus transitions.

By default, an `HNavigable` object does not have any gain focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

**Specified By:**

`setGainFocusSound(HSound)` in interface `HNavigable`

**Parameters:**

`sound` - the sound to be played, when the component gains focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

**setLoseFocusSound(HSound)**

```
public void setLoseFocusSound(HSound sound)
```

Associate a sound with losing focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_LOST`. This sound will start to be played when an object implementing this interface loses focus. It is not guaranteed to be played to completion. It is implementation dependent whether and when this sound will be truncated by any gain focus sound played by the next object to gain focus.

By default, an `HNavigable` object does not have any lose focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

**Specified By:**

`setLoseFocusSound(HSound)` in interface `HNavigable`

**Parameters:**

`sound` - the sound to be played, when the component loses focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

**setMove(int, HNavigable)**

```
public void setMove(int keyCode, HNavigable target)
```

Defines the navigation path from the current `HNavigable` to another `HNavigable` when a particular key is pressed.

Note that `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

**Specified By:**

`setMove(int, HNavigable)` in interface `HNavigable`

**Parameters:**

`keyCode` - The key code of the pressed key. Any numerical keycode is allowed, but the platform may not be able to generate all keycodes. Application authors should only use keys for which `HRcCapabilities.isSupported()` or `HKeyCapabilities.isSupported()` returns true.

`target` - The target `HNavigable` object that should be navigated to. If a target is to be removed from a particular navigation path, then `null` should be specified.

# org.havi.ui HImageEffectMatte

## Syntax

```
public class HImageEffectMatte implements HMatte, HAnimateEffect
```

```
java.lang.Object
|
+--org.havi.ui.HImageEffectMatte
```

## All Implemented Interfaces:

```
HAnimateEffect, HMatte
```

## Description

The `HImageEffectMatte` class represents a matte that varies over both space and time, it is specified as a sequence of image masks.

The data for any `HImageEffectMatte` may be changed "on the fly" using the `setMatteData (Image[ ] )` method. However, some implementations may be asynchronously referencing their content (i.e. through a separate implementation-specific animation thread). Therefore the following restrictions apply to the `setMatteData (Image[ ] )` method:

- The method must be synchronized with any implementation-specific animation thread such that content cannot be changed while a different thread is using it.
- If the animation was running the method should stop the animation in a synchronized manner before changing content.
- The method should reset the animation to a starting position defined by the current play mode. The repeat count of the animation should be reset to 0.
- If the animation was running the method should start the animation.

The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
data	The transparency data for this image effect matte.	null (the matte should be treated as being spatially and temporally unvarying and opaque)	<code>setMatteData (Image[ ] )</code>	<code>getMatteData ()</code>

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
The initial piece of content to be presented, i.e. its position in the content array.	0	<code>setPosition (int)</code>	<code>getPosition ()</code>
By default the animation should be stopped. Hence, to start the animation its start method must be explicitly invoked. This mechanism allows for animations that are programmatically controlled, e.g. via the <code>setPosition</code> method.	"stopped"	<code>start () / stop ()</code>	<code>isAnimated ()</code>

The pixel offset for each image within the `HImageEffectMatte`, relative to the top, left corner of its associated component.

A `java.awt.Point (0, 0)`

`setOffset-`  
`set (Point,`  
`int)`  
`getOffset (i`  
`nt)`

## Constructors

---

### **HImageEffectMatte()**

```
public HImageEffectMatte()
```

Creates an `HImageEffectMatte` object. See the class description for details of constructor parameters and default values.

---

### **HImageEffectMatte(Image[])**

```
public HImageEffectMatte(java.awt.Image[] data)
```

Creates an `HImageEffectMatte` object. See the class description for details of constructor parameters and default values.

## Methods

---

### **getDelay()**

```
public int getDelay()
```

Gets the presentation delay for this `HImageEffectMatte`.

**Specified By:**

`getDelay()` in interface `HAnimateEffect`

**Returns:**

the presentation delay in units of 0.1 seconds.

---

### **getMatteData()**

```
public java.awt.Image[] getMatteData()
```

Returns the data used for this matte.

**Returns:**

the data used for this matte (an array of images) or null if no matte data has been set.

---

### **getOffset(int)**

```
public java.awt.Point getOffset(int index)
```

Get the offset of a specified frame of the matte relative to its component in pixels.

**Parameters:**

`index` - the zero-index to the data for which the offset should be recovered. If `index` is not valid for this matte an `IndexOutOfBoundsException` exception is thrown.

**Returns:**

the offset of the specified frame of the matte relative to its component in pixels (as a Point)

---

### getPlayMode()

```
public int getPlayMode()
```

Gets the playing mode for this `HImageEffectMatte` .

**Specified By:**

`getPlayMode()` in interface `HAnimateEffect`

**Returns:**

the play mode for this `HImageEffectMatte` .

---

### getPosition()

```
public int getPosition()
```

Get the current index into the content array which this `HImageEffectMatte` is using to display content.

**Specified By:**

`getPosition()` in interface `HAnimateEffect`

**Returns:**

the index of the content currently being displayed, in the range  $0 \leq \text{index} < \text{length}$

---

### getRepeatCount()

```
public int getRepeatCount()
```

Gets the number of times that this `HImageEffectMatte` is to be played. Note that this method does *not* return the number of repeats that are remaining to be played.

**Specified By:**

`getRepeatCount()` in interface `HAnimateEffect`

**Returns:**

the total number of times that an `HImageEffectMatte` is to be played. The returned value shall be greater than zero, or `REPEAT_INFINITE` .

---

### isAnimated()

```
public boolean isAnimated()
```

This method indicates the animation (running) state of the `HImageEffectMatte` .

**Specified By:**

`isAnimated()` in interface `HAnimateEffect`

**Returns:**

`true` if this `HImageEffectMatte` is running, i.e. the `start` method has been invoked - `false` otherwise.

---

### setDelay(int)

```
public void setDelay(int count)
```

Sets the delay between the presentation of successive pieces of content (frames).

---

After calling `setDelay(int)` on a currently playing `HImageEffectMatte`, there is no guarantee that one or more frames will not be displayed using the previous delay until the new delay value takes effect.

**Specified By:**

`setDelay(int)` in interface `HAnimateEffect`

**Parameters:**

`count` - the content presentation delay in units of 0.1 seconds duration. If `count` is less than one "unit", then it shall be treated as if it were a delay of one "unit", i.e. 0.1 seconds.

### **setMatteData(Image[])**

```
public void setMatteData(java.awt.Image[] data)
```

Sets the data for this matte (an array of images). Any previously set data is replaced. If this method is called when the animation is running the data is changed immediately and the current animation position is reset according to the active play mode. The changes affect the animation immediately.

Note that if the size of an image is smaller than the size of the component to which the matte is applied, the empty space behaves as if it were an opaque flat matte of value 1.0. By default images are aligned at the top left corner of the component. This can be changed with the `setOffset(Point, int)` method

**Parameters:**

`data` - the data for this matte. Specify a null object to remove the associated data for this matte. If the length of the data array is zero, an `IllegalArgumentException` is thrown. If an element of the data array is null, or an image referred to is still being loaded, then that image will be skipped when the animation is playing.

### **setOffset(Point, int)**

```
public void setOffset(java.awt.Point p, int index)
```

Set the offset of a specified frame of the matte relative to its component in pixels.

**Parameters:**

`p` - the offset of the specified frame of the matte relative to the top left corner of its component in pixels. If `p` is null a `NullPointerException` is thrown.

`index` - the zero-index to the data for which the offset should be applied. If `index` is not valid for this matte an `IndexOutOfBoundsException` exception is thrown.

### **setPlayMode(int)**

```
public void setPlayMode(int mode)
```

Sets the playing mode for this `HImageEffectMatte`. If the animation is already running a call to `setPlayMode` will change the current value and affect the animation immediately. The position of the animation is unchanged.

**Specified By:**

`setPlayMode(int)` in interface `HAnimateEffect`

**Parameters:**

`mode` - the play mode for this `HImageEffectMatte`, which must be either `PLAY_ALTERNATING` or `PLAY_REPEATING`.

### **setPosition(int)**

```
public void setPosition(int position)
```

Set this `HImageEffectMatte` to display the content at the specified position. If the animation is already running a call to `setPosition` will change the current value and affect the animation immediately.

**Specified By:**

```
setPosition(int) in interface HAnimateEffect
```

**Parameters:**

`position` - an index into the content array which specifies the next piece of content to be displayed. If `position` is less than 0, then the array element at index 0 is displayed, if `position` is greater than or equal to the length of the content array, then the array element at index `[length-1]` will be used.

---

**setRepeatCount(int)**

```
public void setRepeatCount(int count)
```

Sets the number of times that this `HImageEffectMatte` should be played. If the animation is already running a call to `setRepeatCount` will change the current value and reset the current number of repeats to 0, affecting the animation immediately.

**Specified By:**

```
setRepeatCount(int) in interface HAnimateEffect
```

**Parameters:**

`count` - the number of times that an `HImageEffectMatte` should be played. Valid values of the repeat count are one or more, and `REPEAT_INFINITE`.

---

**start()**

```
public void start()
```

This method starts this `HImageEffectMatte` playing. If `start` is called when the animation is already running it resets the animation according to the current play mode, as returned by `getPlayMode()`.

**Specified By:**

```
start() in interface HAnimateEffect
```

---

**stop()**

```
public void stop()
```

This method indicates that the running `HImageEffectMatte` should be stopped. After calling this method, there is no guarantee that one or more frames will not be displayed before the animation actually stops playing. If the animation is already stopped further calls to `stop` have no effect.

**Specified By:**

```
stop() in interface HAnimateEffect
```



# org.havi.ui HImageHints

## Syntax

```
public class HImageHints
    java.lang.Object
    |
    +--org.havi.ui.HImageHints
```

## Description

The `HImageHints` object allows an application to pass hints to the system how best to tailor an image to match a (possibly) restricted `HGraphicsConfiguration`. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
-----------	-------------	---------------	------------	------------

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

The image type.	<code>NATURAL_IMAGE</code>	---	---
-----------------	----------------------------	-----	-----

## Fields

---

### BUSINESS\_GRAPHICS

```
public static final int BUSINESS_GRAPHICS
```

The image is business graphics, with strong, well-defined, blocks of solid color, etc. Not suitable for dithering, suitable for nearest color matching.

---

### CARTOON

```
public static final int CARTOON
```

The image is a cartoon, with strong, well-defined, blocks of solid color, etc. Not suitable for dithering, suitable for nearest color matching.

---

### LINE\_ART

```
public static final int LINE_ART
```

The image is a two-tone lineart, with colors varying between foreground and background, etc. Not suitable for dithering. Possibly suitable for color-map adjustment, etc., if applicable.

---

## NATURAL\_IMAGE

```
public static final int NATURAL_IMAGE
```

The image is a "natural" scene, with subtle gradations of color, etc. Suitable for dithering.

## Constructors

---

### HImageHints()

```
public HImageHints()
```

Creates an HImageHints object. See the class description for details of constructor parameters and default values.

## Methods

---

### getType()

```
public int getType()
```

Get the expected type of the image being loaded.

---

### setType(int)

```
public void setType(int type)
```

Set the expected type of the image being loaded.

**Parameters:**

`type` - the expected type of image

# org.havi.ui HImageMatte

## Syntax

```
public class HImageMatte implements HMatte
```

```
java.lang.Object
|
+--org.havi.ui.HImageMatte
```

## All Implemented Interfaces:

[HMatte](#)

## Description

The [HImageMatte](#) class represents a matte that varies over space but is constant over time, it can be specified by an "image mask" (a single channel image) where the pixels indicate matte transparency. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
data	The transparency value for this image matte.	null (the matte should be treated as being spatially unvarying and opaque)	<a href="#">setMatteData(Image)</a>	<a href="#">getMatteData()</a>

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
The pixel offset for the image matte, relative to the top, left corner of its associated component.	A <a href="#">java.awt.Point</a> (0, 0)	<a href="#">setOffset(Point)</a>	<a href="#">getOffset()</a>

## Constructors

### HImageMatte()

```
public HImageMatte()
```

Creates an [HImageMatte](#) object. See the class description for details of constructor parameters and default values.

### HImageMatte(Image)

```
public HImageMatte(java.awt.Image data)
```

Creates an [HImageMatte](#) object. See the class description for details of constructor parameters and default values.

## Methods

---

### getMatteData()

```
public java.awt.Image getMatteData()
```

Returns the data used for this matte.

**Returns:**

the data used for this matte (an image) or null if no matte data has been set.

---

### getOffset()

```
public java.awt.Point getOffset()
```

Get the offset of the matte relative to its component in pixels.

**Returns:**

the offset of the specified frame of the matte relative to its component in pixels (as a Point)

---

### setMatteData(Image)

```
public void setMatteData(java.awt.Image data)
```

Sets the data for this matte. Any previously set data is replaced.

**Parameters:**

`data` - the data for this matte. Specify a null object to remove the associated data for this matte.

---

### setOffset(Point)

```
public void setOffset(java.awt.Point p)
```

Set the offset of the matte relative to its component in pixels.

**Parameters:**

`p` - the offset of the matte relative to its component in pixels. If `p` is null a `NullPointerException` is thrown.

# org.havi.ui HInvalidLookException

## Syntax

```
public class HInvalidLookException extends HUIException
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--HUIException
            |
            +--org.havi.ui.HInvalidLookException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

An HInvalidLookException is an exception that is thrown when a particular HLook is not compatible with the HVisible component it has been associated with. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
message	Message explaining why the exception was thrown	null	-	java.lang.Throwable#getMessage

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Constructors

### HInvalidLookException()

```
public HInvalidLookException()
```

Creates an HInvalidLookException object. See the class description for details of constructor parameters and default values.

### HInvalidLookException(String)

```
public HInvalidLookException(java.lang.String message)
```

Creates an HInvalidLookException object with a specified reason string.

#### Parameters:

message - the reason why the exception was raised

# org.havi.ui

# HItemValue

## Syntax

```
public interface HItemValue extends HNavigable, HSelectionInputPreferred
```

## All Superinterfaces:

```
HNavigable, HNavigationInputPreferred, HOrientable, HSelectionInputPreferred
```

## All Known Implementing Classes:

```
HListGroup
```

## Description

This interface is implemented by all HAVi UI components which have some form of selectable content (e.g. a list group).

### Event Behavior

Subclasses of `HComponent` which implement `HItemValue` must respond to `HFocusEvent` and `HItemEvent` events.

Applications should assume that classes which implement `HItemValue` can generate events of the types `HFocusEvent` and `HItemEvent` in response to other types of input event.

An application may add one or more `HItemListener` listeners to the component. The `selectionChanged(HItemEvent)` method of the `HItemListener` is invoked whenever the selection managed by the `HItemValue` is changed.

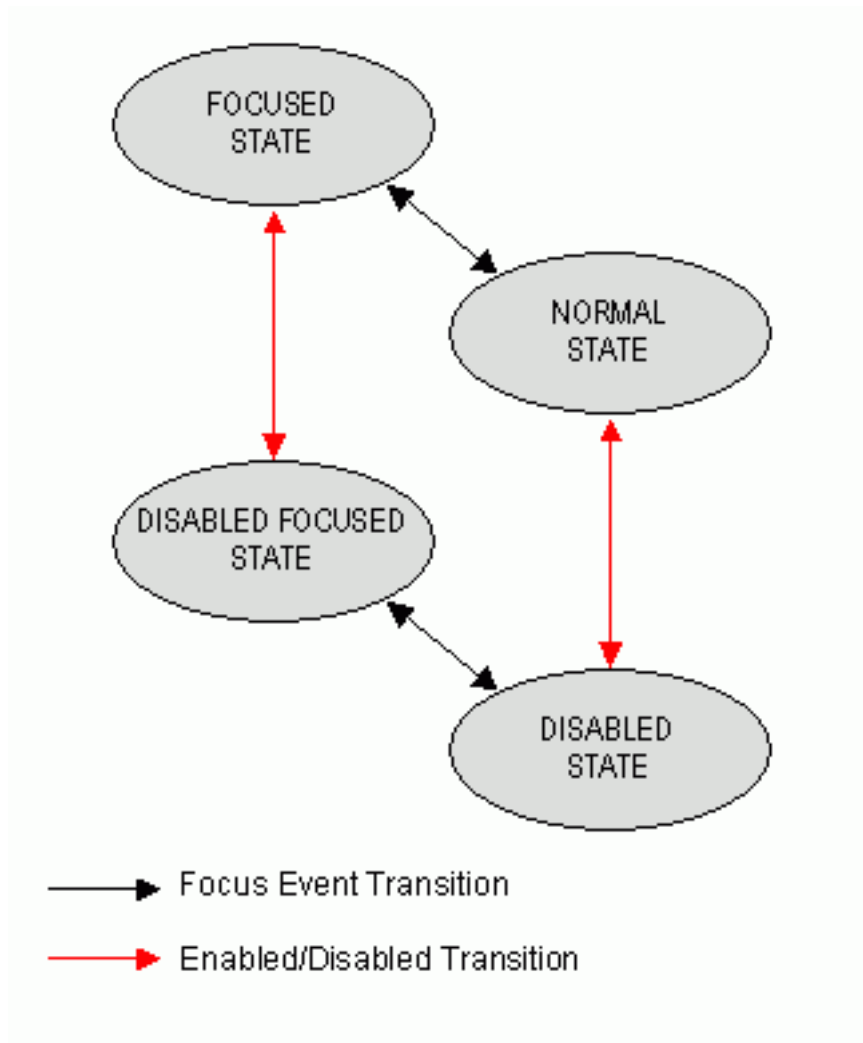
HAVi item events are discussed in detail in the `HSelectionInputPreferred` interface description.

### Interaction States

The following interaction states are valid for this `HItemValue` component:

- `NORMAL_STATE`
- `FOCUSED_STATE`
- `DISABLED_STATE`
- `DISABLED_FOCUSED_STATE`

The state machine diagram below shows the valid state transitions for an `HItemValue` component.



### Platform Classes

The following HAVi platform classes implement or inherit the `HItemValue` interface. These classes shall all generate both `HFocusEvent` and `HItemEvent` events in addition to any other events specified in the respective class descriptions.

- `HListGroup`

### See Also:

`HNavigable`, `HOrientable`, `HSelectionInputPreferred`, `HItemEvent`, `HItemListener`

## Methods

---

### **addItemListener(HItemListener)**

```
public void addItemListener(HItemListener l)
```

Adds the specified `HItemListener` to receive `HItemEvent` sent from this object. If the listener has already been added further calls will add further references to the listener, which will then receive multiple copies of a single event.

**Parameters:**

`l` - the `HItemListener` to be notified.

---

**getSelectionSound()**

```
public HSound getSelectionSound()
```

Get the sound to be played when the selection changes.

**Returns:**

The sound played when the selection changes

---

**removeItemListener(HItemListener)**

```
public void removeItemListener(HItemListener l)
```

Removes the specified `HItemListener` so that it no longer receives `HItemEvent` from this object. If the specified listener is not registered, the method has no effect. If multiple references to a single listener have been registered it should be noted that this method will only remove one reference per call.

**Parameters:**

`l` - the `HItemListener` to be removed from notification.

---

**setSelectionSound(HSound)**

```
public void setSelectionSound(HSound sound)
```

Associate a sound to be played when the selection is modified. The sound is played irrespective of whether an `HItemEvent` is sent to one or more listeners.

**Parameters:**

`sound` - the sound to be played, when the selection is modified. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.



# org.havi.ui HKeyboardInputPreferred

## Syntax

```
public interface HKeyboardInputPreferred
```

## All Known Subinterfaces:

```
    HTextValue
```

## Description

A component which implements `HKeyboardInputPreferred` indicates that this component expects to receive both `HKeyEvent` and `HTextEvent` input events.

All interoperable implementations of the `HKeyboardInputPreferred` interface must extend `HComponent`.

The set of characters which the component expects to receive via `HKeyEvent` events is defined by the return code from the `getType()` method.

When this component has focus, platforms without a physical means of generating key events with the desired range of characters will provide another means for keyboard entry e.g. by offering an on-screen "virtual" keyboard. Applications can query the system about the support of specific keyCodes through the `isSupported(int)` method.

Note that the `java.awt.Component` method `isFocusTraversable` should always return true for a `java.awt.Component` implementing this interface.

## Fields

---

### INPUT\_ALPHA

```
public static final int INPUT_ALPHA
```

This constant indicates that the component only requires alphanumeric input, as determined by the `java.lang.Character` `isLetter` method.

---

### INPUT\_ALPHANUMERIC

```
public static final int INPUT_ALPHANUMERIC
```

This constant indicates that the component only requires alphanumeric input, as determined by the `java.lang.Character` `isLetterOrDigit` method.

---

### INPUT\_ANY

```
public static final int INPUT_ANY
```

Indicates that the component requires any possible character as input, as determined by the `java.lang.Character` `isDefined` method.

---

## INPUT\_CUSTOMIZED

```
public static final int INPUT_CUSTOMIZED
```

Indicates that the component requires as input the characters present in the array returned from the `getValidInput()` method.

---

## INPUT\_NUMERIC

```
public static final int INPUT_NUMERIC
```

This constant indicates that the component only requires alphanumeric input, as determined by the `java.lang.Character isDigit` method.

## Methods

---

### getEditMode()

```
public boolean getEditMode()
```

Get the editing mode for this `HKeyboardInputPreferred`. If the returned value is `true` the component is in edit mode, and its textual content may be changed through user interaction such as keyboard events.

The component is switched into and out of edit mode on receiving `TEXT_START_CHANGE` and `TEXT_END_CHANGE` events.

**Returns:**

`true` if this component is in edit mode, `false` otherwise.

---

### getType()

```
public int getType()
```

Retrieve the desired input type for this component. This value should be set to indicate to the system which input keys are required by this component.

**Returns:**

one of `INPUT_ALPHANUMERIC`, `INPUT_NUMERIC`, `INPUT_ALPHA`, or `INPUT_CUSTOMIZED`.

---

### getValidInput()

```
public char[] getValidInput()
```

Retrieve the customized input character range. The return value of this method should reflect the range of input keys which the component wishes to see, should `getType()` return `INPUT_CUSTOMIZED`. This method may return `null` if and only if it can guarantee that customized input is *never* required.

If customized input can be used but no specific customized input is set, this method should return the same range as the `INPUT_ALPHANUMERIC` input type, i.e. that range of characters for which the `java.lang.Character isLetterOrDigit` returns `true`.

**Returns:**

an array containing the characters which this component expects the platform to provide, or `null` to indicate that customized characters are not used.

---

**processHKeyEvent(HKeyEvent)**

```
public void processHKeyEvent(HKeyEvent evt)
```

Process an `HKeyEvent` sent to this `HKeyboardInputPreferred`.

**Parameters:**

`evt` - the `HKeyEvent` to process.

---

**processHTextEvent(HTextEvent)**

```
public void processHTextEvent(HTextEvent evt)
```

Process an `HTextEvent` sent to this `HKeyboardInputPreferred`.

**Parameters:**

`evt` - the `HTextEvent` to process.

---

**setEditMode(boolean)**

```
public void setEditMode(boolean edit)
```

Set the editing mode for this `HKeyboardInputPreferred`.

This method is provided for the convenience of component implementors. Interoperable applications shall not call this method. It cannot be made protected because interfaces cannot have protected methods.

**Parameters:**

`edit` - true to switch this component into edit mode, false otherwise.

**See Also:**

`getEditMode()`

# org.havi.ui HListElement

## Syntax

```
public class HListElement
    java.lang.Object
    |
    +--org.havi.ui.HListElement
```

## Description

`HListElement` is a holder for content used with the `HListGroup` component. It must contain a text string, and may also contain a single graphical image.

Applications should **not** directly manipulate `HListElement` objects. They are intended to be used in conjunction with an `HListGroup` which maintains a list of them, and is responsible for their rendering via the `HListGroupLook` class. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
label	text content of this item	no default	-	<code>getLabel()</code>
icon	image content of this item	null	-	<code>getIcon()</code>

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## See Also:

[HListGroup](#), [HListGroupLook](#)

## Constructors

### **HListElement(Image, String)**

```
public HListElement(java.awt.Image icon, java.lang.String label)
```

Creates an `HListElement` object. See the class description for details of constructor parameters and default values.

#### **Parameters:**

- `icon` - The icon for this `HListElement`.
- `label` - The label for this `HListElement`.

---

## HListElement(String)

```
public HListElement(java.lang.String label)
```

Creates an HListElement object. See the class description for details of constructor parameters and default values.

**Parameters:**

`label` - The label for this HListElement.

## Methods

---

### getIcon()

```
public java.awt.Image getIcon()
```

Retrieve the icon for this HListElement.

**Returns:**

the graphical icon for this HListElement, or `null` if no icon was set.

---

### getLabel()

```
public java.lang.String getLabel()
```

Retrieve the label for this HListElement.

**Returns:**

the text label for this HListElement.

---

### setIcon(Image)

```
public void setIcon(java.awt.Image icon)
```

Set the icon for this HListElement. If `icon` is `null`, the HListElement will be in the same state as if no icon was set.

**Parameters:**

`icon` - The icon for this HListElement.

---

### setLabel(String)

```
public void setLabel(java.lang.String label)
```

Set the label for this HListElement.

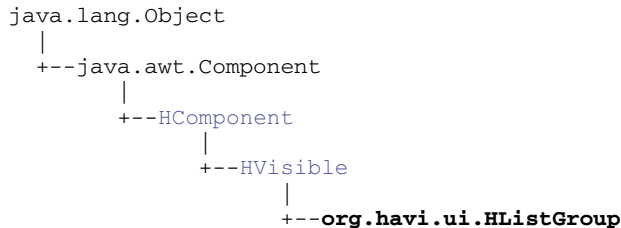
**Parameters:**

`label` - The label for this HListElement.

# org.havi.ui HListGroup

## Syntax

```
public class HListGroup extends HVisible implements HItemValue
```



## All Implemented Interfaces:

HItemValue, HMatteLayer, HNavigable, HNavigationInputPreferred, HOrientable, HSelectionInputPreferred, HState, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, TestOpacity

## Description

The `HListGroup` is a user interface component representing a list of selectable items (`HListElement`) which contain static read-only graphical and / or textual content.

This component can be navigated to, i.e. it can have the input focus. It also responds to `HItemEvent` events as follows:

- When the component has focus, sending an `ITEM_START_CHANGE` event to the component causes it to enter selection mode.
- When the component has focus, sending an `ITEM_END_CHANGE` event to the component causes it to leave selection mode.
- When the component has focus and is in selection mode, the current item can be set by sending `ITEM_SET_CURRENT`, `ITEM_SET_PREVIOUS` and `ITEM_SET_NEXT` events to the component.
- When the component has focus and is in selection mode, sending an `ITEM_TOGGLE_SELECTED` event causes the current item to be toggled between a selected and unselected state.
- Irrespective of focus and selection mode, sending an `ITEM_SELECTION_CLEARED` event to the component causes the current selection set to be cleared. The position of the current item is unchanged.

`HListGroup` has the following properties which make it slightly different from the other platform components.

- uses the `HTextLayoutManager` to render text from the elements. `HListGroup` is not required to respect the default horizontal and vertical content alignments specified by `HVisible`. For `HListGroup` these defaults are implementation specific. Application programmers who require a specific alignment policy must therefore make explicit calls to `setHorizontalAlignment(int)` and `setVerticalAlignment(int)` to enforce the alignments required.
- the resize mode as defined by `HVisible` determines how the icons are scaled.
- focus traversal applies to the entire list component. The elements in the list are not components in their own right and never receive focus. The concept of the current element is handled through `HItemEvent` events.
- for the purpose of layout management of the `HListGroup` component, the following constraints are applied:
  5. the minimum size is the size to present one element or an implementation specific minimum (32 x 32 for example) if no elements are present.
  6. the preferred size is that set by `setDefaultSize(Dimension)` rounded down to the nearest element (minimum of one) or the size required to present 5 elements if a default size is not set.
  7. the maximum size is that required to present all elements.

By default this component uses the `HListGroupLook` class to render itself. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
width	width of this component in pixels (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
height	height of this component in pixels (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte ( <code>HMatte</code> ).	none (i.e. <code>getMatte()</code> returns null)	<code>setMatte (HMatte )</code>	<code>getMatte ()</code>
The text layout manager responsible for text formatting.	An <code>HDefaultTextLayoutManager</code> object.	<code>setTextLayoutManager (HTextLayoutManager)</code>	<code>getTextLayoutManager ()</code>
The background painting mode	<code>NO_BACKGROUND_FILL</code>	<code>setBackgroundMode (int)</code>	<code>getBackgroundMode ()</code>
The default preferred size	not set (i.e. null) unless specified by width and height parameters	<code>setDefaultSize (Dimension)</code>	<code>getDefaultSize ()</code>
The horizontal content alignment	<code>HALIGN_CENTER</code>	<code>setHorizontalAlignment (int)</code>	<code>getHorizontalAlignment ()</code>
The vertical content alignment	<code>VALIGN_CENTER</code>	<code>setVerticalAlignment (int)</code>	<code>getVerticalAlignment ()</code>
The content scaling mode	<code>RESIZE_NONE</code>	<code>setResizeMode (int)</code>	<code>getResizeMode ()</code>
The default "look" for this class.	A platform specific <code>HListGroupLook</code>	<code>setDefaultLook (HListGroupLook)</code>	<code>getDefaultLook ()</code>

The "look" for this object.	The <code>HListGroupLook</code> returned from <code>HListGroup.getDefaultLook</code> when this object was created.	<code>setLook(HLook)</code>	<code>getLook()</code>
The gain focus sound.	<code>null</code>	<code>setGainFocusSound(HSound)</code>	<code>getGainFocusSound()</code>
The lose focus sound.	<code>null</code>	<code>setLoseFocusSound(HSound)</code>	<code>getLoseFocusSound()</code>
The selection sound.	<code>null</code>	<code>setSelectionSound(HSound)</code>	<code>getSelectionSound()</code>

**See Also:**

`HListElement`, `HListGroupLook`, `HNavigable`, `HItemValue`

**Fields****ADD\_INDEX\_END**

```
public static final int ADD_INDEX_END
```

A constant for use with `addItem(HListElement, int)` and `addItems(HListElement[], int)` which specifies that the new items should be appended to the end of the list.

**DEFAULT\_ICON\_HEIGHT**

```
public static final int DEFAULT_ICON_HEIGHT
```

A constant for use with `setIconSize(Dimension)` and `getIconSize()`. When no call to `setIconSize(Dimension)` has been made then `getIconSize()` will return this value for its default height. The default icon height for all orientations is implementation specific.

**DEFAULT\_ICON\_WIDTH**

```
public static final int DEFAULT_ICON_WIDTH
```

A constant for use with `setIconSize(Dimension)` and `getIconSize()`. When no call to `setIconSize(Dimension)` has been made then `getIconSize()` will return this value for its default width. The default icon width for all orientations is implementation specific.

**DEFAULT\_LABEL\_HEIGHT**

```
public static final int DEFAULT_LABEL_HEIGHT
```

A constant for use with `setLabelSize(Dimension)` and `getLabelSize()`. When no call to `setLabelSize(Dimension)` has been made then `getLabelSize()` will return this value for its default height. The default label height for all orientations is the current font height.



---

**DEFAULT\_LABEL\_WIDTH**

```
public static final int DEFAULT_LABEL_WIDTH
```

A constant for use with `setLabelSize(Dimension)` and `getLabelSize()`. When no call to `setLabelSize(Dimension)` has been made then `getLabelSize()` will return this value for its default width. The default label width for all orientations is implementation specific.

---

**ITEM\_NOT\_FOUND**

```
public static final int ITEM_NOT_FOUND
```

A constant which may be returned from `getIndex(HListElement)` if the requested element is not found in the content.

## Constructors

---

**HListGroup()**

```
public HListGroup()
```

Creates an `HListGroup` object. See the class description for details of constructor parameters and default values.

---

**HListGroup(HListElement[])**

```
public HListGroup(HListElement[] items)
```

Creates an `HListGroup` object. See the class description for details of constructor parameters and default values.

---

**HListGroup(HListElement[], int, int, int, int)**

```
public HListGroup(HListElement[] items, int x, int y, int width, int height)
```

Creates an `HListGroup` object. See the class description for details of constructor parameters and default values.

## Methods

---

**addItem(HListElement, int)**

```
public void addItem(HListElement item, int index)
```

Add an item to this `HListGroup`. The item is inserted after the specified `HListElement` in the content list. If no content exists a new content list is created to contain the new item and the value of the `index` parameter is ignored.

If the act of adding a new item causes the current active item index to change, an `HAdjustmentEvent` shall be sent.

Note that items are stored in the content list by reference, they are not copied.

**Parameters:**

`item` - the item to add.

`index` - the index of the currently existing item which the new items should be placed after, or `ADD_INDEX_END` to append the new items to the end of the list. If this value is not a valid item index for this list a `java.lang.IndexOutOfBoundsException` shall be thrown.

**addItemListener(HItemListener)**

```
public void addItemListener(HItemListener l)
```

Adds the specified `HItemListener` to receive `HItemEvent` sent from this object. If the listener has already been added further calls will add further references to the listener, which will then receive multiple copies of a single event.

**Specified By:**

`addItemListener(HItemListener)` in interface `HItemValue`

**Parameters:**

`l` - the `HItemListener` to be notified.

**addItem(HListElement[], int)**

```
public void addItem(HListElement[] items, int index)
```

Add an array of items to this `HListGroup`. The items are inserted in the same order as they are in the array after the item specified. If no content exists a new content list is created to contain the new items and the value of the `index` parameter is ignored.

If the act of adding a new item causes the current active item index to change, an `HAdjustmentEvent` shall be sent.

Note that items are stored in the content list by reference, they are not copied.

**Parameters:**

`item` - the item to add.

`index` - the index of the currently existing item which the new items should be placed after, or `ADD_INDEX_END` to append the new items to the end of the list. If this value is not a valid item index for this list a `java.lang.IndexOutOfBoundsException` shall be thrown.

**clearSelection()**

```
public void clearSelection()
```

Destroy the selection. This method deselects any selected `HListElement`, but does not remove them from the `HListGroup`. After calling this method calls to `getSelection()` shall return `null` until a new selection is made.

If the selection was not already empty, an `HItemEvent` shall be sent to any registered listeners.

**See Also:**

`getSelection()`

**getCurrentIndex()**

```
public int getCurrentIndex()
```

Retrieve the current active item index, if one is currently chosen. The current index is the index of the `HListElement` which would be selected or deselected should the user action the `HListGroup`. If there is no current element or there is no content set this method shall return `ITEM_NOT_FOUND`

**Returns:**

the current item index, or `ITEM_NOT_FOUND` if no such item exists.

---

**getCurrentItem()**

```
public HListElement getCurrentItem()
```

Retrieve the current active item, if one has been chosen. The current item is the `HListElement` which would be selected or deselected should the user action the `HListGroup`. If there is no current item or there is no content set this method shall return `null`

**Returns:**

the current item, or `null` if no such item exists.

---

**getDefaultLook()**

```
public static HListGroupLook getDefaultLook()
```

Returns the currently set default `HLook` for `HListGroup` components.

**Returns:**

The `HLook` that is used by default when creating a new `HListGroup` component.

---

**getGainFocusSound()**

```
public HSound getGainFocusSound()
```

Get the sound associated with the gain focus event.

**Specified By:**

`getGainFocusSound()` in interface `HNavigable`

**Returns:**

The sound played when the component gains focus. If no sound is associated with gaining focus, then `null` shall be returned.

---

**getIconSize()**

```
public java.awt.Dimension getIconSize()
```

Retrieve the icon size for this `HListGroup`. This size is the desired size of the area into which the `HListGroupLook` should render any image content of the `HListElement`.

**Returns:**

the icon size. If no size has been set then this method shall return `new Dimension(DEFAULT_ICON_WIDTH, DEFAULT_ICON_HEIGHT)`.

---

**getIndex(HListElement)**

```
public int getIndex(HListElement item)
```

Retrieve the index position of an item in the content list.

**Parameters:**

`item` - the item to retrieve the index for.

**Returns:**

the index of the given `HListElement`, or `ITEM_NOT_FOUND` if no such element exists.

---

---

**getItem(int)**

```
public HListElement getItem(int index)
```

Retrieve an item from the content list by index.

**Parameters:**

`index` - the index of the item to retrieve. If this parameter is negative a `java.lang.IllegalArgumentException` shall be thrown.

**Returns:**

the `HListElement` at the given index, or `null` if no such element exists.

---

**getLabelSize()**

```
public java.awt.Dimension getLabelSize()
```

Retrieve the label size for this `HListGroup`. This size is the desired size of the area into which the `HListGroupLook` should render any textual content of the `HListElement`.

**Returns:**

the label size. If no size has been set then this method shall return `new Dimension(DEFAULT_LABEL_WIDTH, DEFAULT_LABEL_HEIGHT)`.

---

**getListContent()**

```
public HListElement[] getListContent()
```

Retrieve the list content for this `HListGroup`.

**Returns:**

the list content or `null` if no content has been set.

---

**getLoseFocusSound()**

```
public HSound getLoseFocusSound()
```

Get the sound associated with the lost focus event.

**Specified By:**

`getLoseFocusSound()` in interface `HNavigable`

**Returns:**

The sound played when the component loses focus. If no sound is associated with losing focus, then `null` shall be returned.

---

**getMove(int)**

```
public HNavigable getMove(int keyCode)
```

Provides the `HNavigable` object that is navigated to when a particular key is pressed.

**Specified By:**

`getMove(int)` in interface `HNavigable`

**Parameters:**

`keyCode` - The key code of the pressed key.

**Returns:**

Returns the `HNavigable` object, or if no `HNavigable` is associated with the `keyCode` then returns `null`.

---

---

**getMultiSelection()**

```
public boolean getMultiSelection()
```

Return the multiple selection mode currently active for this `HListGroup`. Multiple selection mode means that there may be more than one `HListElement` selected at a time.

**Returns:**

`true` if multiple selections are permitted, `false` otherwise.

---

**getNavigationKeys()**

```
public int[] getNavigationKeys()
```

Retrieve the set of key codes which this component maps to navigation targets.

**Specified By:**

`getNavigationKeys()` in interface `HNavigationInputPreferred`

**Returns:**

an array of key codes, or `null` if no navigation targets are set on this component.

---

**getNumItems()**

```
public int getNumItems()
```

Retrieve the number of items in the content list.

**Returns:**

the number of items in the content list, or 0 if no content has been set.

---

**getNumSelected()**

```
public int getNumSelected()
```

Return the number of items which would be in the selection, if the `getSelection()` method were called at this time.

**Returns:**

the number of selected items.

---

**getOrientation()**

```
public int getOrientation()
```

Retrieve the orientation of the `HListGroup`. The orientation controls how an associated `HLook` lays out the component and affects the visual behavior of the `HAdjustmentEvent` and `HItemEvent` events. For example, the system might use this information to select appropriate key mappings for these events.

**Specified By:**

`getOrientation()` in interface `HOrientable`

**Returns:**

one of `ORIENT_LEFT_TO_RIGHT`, `ORIENT_RIGHT_TO_LEFT`, `ORIENT_TOP_TO_BOTTOM`, or `ORIENT_BOTTOM_TO_TOP`.

---

**getScrollPosition()**

```
public int getScrollPosition()
```

Retrieve the scroll position of the `HListGroup`. The scroll position determines the first `HListElement` to be drawn when the `HListGroupLook` lays out the list.

**Returns:**

the current scroll position.

---

**getSelection()**

```
public HListElement[] getSelection()
```

Get the selection from this `HListGroup`. The selection is defined as that set of `HListElement` which the user has caused to be selected by actioning the `HListGroup`.

**Returns:**

the selection, or `null` if no items are selected. Only items which are currently part of the content for this `HListGroup` may be selected.

---

**getSelectionIndices()**

```
public int[] getSelectionIndices()
```

Get the list of selection indices from this `HListGroup`. The selection is defined as that set of `HListElement` indices which the user has caused to be selected by actioning the `HListGroup`.

**Returns:**

the index selection, or `null` if no items are selected. Only items which are currently part of the content for this `HListGroup` may be selected.

---

**getSelectionMode()**

```
public boolean getSelectionMode()
```

Get the selection mode for this `HListGroup`. If the returned value is `true` the component is in selection mode, and the selection may be changed.

The component is switched into and out of selection mode on receiving `ITEM_START_CHANGE` and `ITEM_END_CHANGE` events.

**Specified By:**

`getSelectionMode()` in interface `HSelectionInputPreferred`

**Returns:**

`true` if this component is in selection mode, `false` otherwise.

---

**getSelectionSound()**

```
public HSound getSelectionSound()
```

Get the sound to be played when the selection changes.

**Specified By:**

`getSelectionSound()` in interface `HItemValue`

**Returns:**

The sound played when the selection changes

---

**isItemSelected(int)**

```
public boolean isItemSelected(int index)
```

Retrieve the selection state of a particular `HListElement`.

---

Note that if multiple selections are not permitted only one `HListElement` may be selected at a time.

**Parameters:**

`index` - the index of the `HListElement` to query. A `java.lang.IllegalArgumentException` shall be thrown if this index is not valid for the `HListGroup`.

**Returns:**

`true` if the given `HListElement` is selected, `false` otherwise.

**isSelected()**

```
public boolean isSelected()
```

Indicates if this component has focus.

**Specified By:**

`isSelected()` in interface `HNavigable`

**Returns:**

`true` if the component has focus, otherwise returns `false`.

**processHFocusEvent(HFocusEvent)**

```
public void processHFocusEvent(HFocusEvent evt)
```

Process an `HFocusEvent` sent to this `HListGroup`.

**Specified By:**

`processHFocusEvent(HFocusEvent)` in interface `HNavigationInputPreferred`

**Parameters:**

`evt` - the `HFocusEvent` to process.

**processHItemEvent(HItemEvent)**

```
public void processHItemEvent(HItemEvent evt)
```

Process an `HItemEvent` sent to this `HListGroup`.

**Specified By:**

`processHItemEvent(HItemEvent)` in interface `HSelectionInputPreferred`

**Parameters:**

`evt` - the `HItemEvent` to process.

**removeAllItems()**

```
public void removeAllItems()
```

Remove all the content. The selection is also destroyed and calls to `getSelection()` shall return `null` until new content and selections are created.

**See Also:**

`getSelection()`

**removeItem(int)**

```
public HListElement removeItem(int index)
```

Remove the `HListElement` at the specified index. The item is also removed from the selection, if any is set. If this was the last item in the selection the entire selection is destroyed and calls to `getSelection()` shall return `null` until new content and selections are created.

If the act of removing an item causes the current active item index to change, an `HAdjustmentEvent` shall be sent.

If the act of removing an item causes the selection to change, an `HItemEvent` shall be sent.

**Parameters:**

`index` - the index of the item to remove.

**Returns:**

the `HListElement` that has been removed or `null` if the index is not valid. No exception is thrown if `index` is not valid.

**See Also:**

`getSelection()`

### **removeItemListener(HItemListener)**

```
public void removeItemListener(HItemListener l)
```

Removes the specified `HItemListener` so that it no longer receives `HItemEvent` from this object. If the specified listener is not registered, the method has no effect. If multiple references to a single listener have been registered it should be noted that this method will only remove one reference per call.

**Specified By:**

`removeItemListener(HItemListener)` in interface `HItemValue`

**Parameters:**

`l` - the `HItemListener` to be removed from notification.

### **setCurrentItem(int)**

```
public boolean setCurrentItem(int index)
```

Set the current active item. The current item is the `HListElement` which would be selected or deselected should the user action the `HListGroup`.

**Parameters:**

`index` - the index of the new current item.

**Returns:**

`true` if the current item was changed, `false` if `index` was not a valid index for this `HListGroup` or the current item was not changed because it is already selected. No exception is thrown if `index` is not valid.

### **setDefaultLook(HListGroupLook)**

```
public static void setDefaultLook(HListGroupLook look)
```

Sets the default `HLook` for further `HListGroup` Components.

**Parameters:**

`look` - The `HLook` that will be used by default when creating a new `HListGroup` component. Note that this parameter may be null, in which case newly created components shall not draw themselves until a non-null look is set using the `setLook(HLook)` method.



---

**setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)**

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
                             HNavigable right)
```

Set the focus control for an `HNavigable` component. Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is a convenience function for application programmers where a standard up, down, left and right focus traversal between components is required.

Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

Note that this API does not prevent the creation of "isolated" `HNavigable` components --- authors should endeavor to avoid confusing the user.

**Specified By:**

```
setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) in
interface HNavigable
```

**Parameters:**

`up` - The `HNavigable` component to move to, when the user generates a `VK_UP` KeyEvent. If there is no `HNavigable` component to move "up" to, then null should be specified.

`down` - The `HNavigable` component to move to, when the user generates a `VK_DOWN` KeyEvent. If there is no `HNavigable` component to move "down" to, then null should be specified.

`left` - The `HNavigable` component to move to, when the user generates a `VK_LEFT` KeyEvent. If there is no `HNavigable` component to move "left" to, then null should be specified.

`right` - The `HNavigable` component to move to, when the user generates a `VK_RIGHT` KeyEvent. If there is no `HNavigable` component to move "right" to, then null should be specified.

---

**setGainFocusSound(HSound)**

```
public void setGainFocusSound(HSound sound)
```

Associate a sound with gaining focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_GAINED`. This sound will start to be played when an object implementing this interface gains focus. It is not guaranteed to be played to completion. If the object implementing this interface loses focus before the audio completes playing, the audio will be truncated. Applications wishing to ensure the audio is always played to completion must implement special logic to slow down the focus transitions.

By default, an `HNavigable` object does not have any gain focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

**Specified By:**

```
setGainFocusSound(HSound) in interface HNavigable
```

**Parameters:**

`sound` - the sound to be played, when the component gains focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

---

**setIconSize(Dimension)**

```
public void setIconSize(java.awt.Dimension size)
```

Set the icon size for this `HListGroup`. This size is the desired size of the area into which the `HListGroupLook` should render any image content of the `HListElement`.

**Parameters:**

`size` - the icon size. If this parameter is `new Dimension(DEFAULT_ICON_SIZE, DEFAULT_ICON_SIZE)` or `null` the `HListGroup` shall revert to using an implementation-specific icon size.

### **setItemSelected(int, boolean)**

```
public void setItemSelected(int index, boolean sel)
```

Set the selection state of a particular `HListElement`.

If a successful call to this method causes the selection to change an `HItemEvent` shall be sent to any registered listeners.

**Parameters:**

`index` - the index of the `HListElement` to alter. A `java.lang.IllegalArgumentException` shall be thrown if this index is not valid for the `HListGroup`.

`sel` - `true` to select the given `HListElement`, `false` otherwise.

### **setLabelSize(Dimension)**

```
public void setLabelSize(java.awt.Dimension size)
```

Set the label size for this `HListGroup`. This size is the desired size of the area into which the `HListGroupLook` should render any textual content of the `HListElement`.

**Parameters:**

`size` - the label size. If this parameter is `new Dimension(DEFAULT_LABEL_WIDTH, DEFAULT_LABEL_HEIGHT)` or `null` the `HListGroup` shall revert to using an implementation-specific label size.

### **setListContent(HListElement[])**

```
public void setListContent(HListElement[] elements)
```

Set the list content for this `HListGroup`. Any existing selection is discarded (which may cause an `HItemEvent` to be generated.)

**Parameters:**

`elements` - the list content. If this parameter is `null` any existing content is removed.

### **setLook(HLook)**

```
public void setLook(HLook hlook)
```

Sets the `HLook` for this component.

**Overrides:**

`setLook(HLook)` in class `HVisible`

**Parameters:**

`hlook` - The `HLook` that is to be used for this component. Note that this parameter may be `null`, in which case the component will not draw itself until a look is set.

**Throws:**

`HInvalidLookAndFeelException` - If the Look is not an `HListGroupLook`.

---

**setLoseFocusSound(HSound)**

```
public void setLoseFocusSound(HSound sound)
```

Associate a sound with losing focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_LOST`. This sound will start to be played when an object implementing this interface loses focus. It is not guaranteed to be played to completion. It is implementation dependent whether and when this sound will be truncated by any gain focus sound played by the next object to gain focus.

By default, an `HNavigable` object does not have any lose focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

**Specified By:**

`setLoseFocusSound(HSound)` in interface `HNavigable`

**Parameters:**

`sound` - the sound to be played, when the component loses focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

---

**setMove(int, HNavigable)**

```
public void setMove(int keyCode, HNavigable target)
```

Defines the navigation path from the current `HNavigable` to another `HNavigable` when a particular key is pressed.

Note that `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

**Specified By:**

`setMove(int, HNavigable)` in interface `HNavigable`

**Parameters:**

`keyCode` - The key code of the pressed key. Any numerical keycode is allowed, but the platform may not be able to generate all keycodes. Application authors should only use keys for which `HRCcapabilities.isSupported()` or `HKeyCapabilities.isSupported()` returns true.

`target` - The target `HNavigable` object that should be navigated to. If a target is to be removed from a particular navigation path, then `null` should be specified.

---

**setMultiSelection(boolean)**

```
public void setMultiSelection(boolean multi)
```

Set the multiple selection mode for this `HListGroup`. Multiple selection mode means that there may be more than one `HListElement` selected at a time.

Note that if the `HListGroup` is switched out of multiple selection mode and more than one item is selected, the selection shall change so that the first of the items is selected and the others are deselected. This will cause an `HItemEvent` to be sent to any registered listeners.

**Parameters:**

`true` - if multiple selections are to be permitted, `false` otherwise.

---

**setOrientation(int)**

```
public void setOrientation(int orient)
```

Set the orientation of the `HListGroup`. The orientation controls how the associated `HLook` lays out the component.

**Specified By:**

`setOrientation(int)` in interface `HOrientable`

**Parameters:**

`orient` - one of `ORIENT_LEFT_TO_RIGHT`, `ORIENT_RIGHT_TO_LEFT`, `ORIENT_TOP_TO_BOTTOM`, or `ORIENT_BOTTOM_TO_TOP`.

---

**setScrollPosition(int)**

```
public void setScrollPosition(int scroll)
```

Set the scroll position of the `HListGroup`. The scroll position determines the first `HListElement` to be drawn when the `HListGroupLook` lays out the list. An `IllegalArgumentException` shall be thrown if `scroll` is not a valid scroll position.

It is an implementation option for `HListGroupLook` to draw elements before this first one, in order to fill the available space.

**Returns:**

the current scroll position, or 0 if no content is set.

---

**setSelectionMode(boolean)**

```
public void setSelectionMode(boolean adjust)
```

Set the selection mode for this `HListGroup`.

This method is provided for the convenience of component implementors. Interoperable applications shall not call this method. It cannot be made protected because interfaces cannot have protected methods.

**Specified By:**

`setSelectionMode(boolean)` in interface `HSelectionInputPreferred`

**Parameters:**

`edit` - true to switch this component into selection mode, false otherwise.

**See Also:**

`getSelectionMode()`

---

**setSelectionSound(HSound)**

```
public void setSelectionSound(HSound sound)
```

Associate a sound to be played when the selection is modified. The sound is played irrespective of whether an `HItemEvent` is sent to one or more listeners.

**Specified By:**

`setSelectionSound(HSound)` in interface `HItemValue`

**Parameters:**

`sound` - the sound to be played, when the selection is modified. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

# org.havi.ui HListGroupLook

## Syntax

```
public class HListGroupLook implements HAdjustableLook
```

```
java.lang.Object
|
+--org.havi.ui.HListGroupLook
```

## All Implemented Interfaces:

```
java.lang.Cloneable, HAdjustableLook, HLook
```

## Description

The `HListGroupLook` class is used by the `HListGroup` component to display both the `HListGroup` itself (potentially including a scrollbar component) and graphical or textual list content held on the `HListGroup`. This look will be provided by the platform and the exact way in which it is rendered will be platform dependent.

The `HListGroupLook` class draws the `HListGroup` and any look-specific borders around the component, and then renders the content set on the `HListGroup`. It uses the `getListContent()` method to determine the content to render. The content of the `HListGroup` does not depend on the interaction state. Borders are not drawn around the content.

`HListGroupLook` should use the following properties of `HListGroup` to lay out and render the `HListElement` content:

Item	Method	Purpose
Orientation	<code>getOrientation()</code>	direction to lay out elements
Content	<code>getListContent()</code>	elements to display
Scroll position	<code>getScrollPosition()</code>	first element to draw
Selection	<code>isItemSelected(int)</code>	mark an element as selected
Current item	<code>getCurrentItem()</code>	highlight an element

`HListGroupLook` should draw a scrollbar as necessary when there are more `HListElement` than can be displayed. It is an implementation option to leave border space between each element. The insets used for the element borders can be retrieved using `getElementInsets()`

Implementations of `HListGroupLook` should use the appropriate methods on `HListGroup` to determine which scaling and alignment modes to use when rendering content. See the class description for `HLook` for more details.

`HListGroupLook` may support scalable graphical content. As a minimum, all implementations must support the `RESIZE_NONE` scaling mode for graphical content, and all alignment modes for text content. However, Note that `HListGroupLook` behaves slightly differently from other HAVi `HLook` classes, as follows.

- Where supported, scaling applies to the icon (graphical content) of each `HListElement`, based on the area

allocated to that `HListElement` rather than the entire area of the `HListGroup`.

- Alignment mode applies to the content of the `HListElement` within the area allocated to that `HListElement` rather than the entire area of the `HListGroup`.

Note that the results of applying the `VALIGN_JUSTIFY` and `HALIGN_JUSTIFY` alignment modes to graphical content are defined to identical to `VALIGN_CENTER` and `HALIGN_CENTER` modes respectively, as justification is meaningless in this context.

This is the default look that is used by `HListGroup`. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Element insets	null	<code>getElementInsets()</code>	---

### See Also:

`HListGroup`, `HListElement`, `HVisible`, `HLook`, `HDefaultTextLayoutManager`

## Constructors

### `HListGroupLook()`

```
public HListGroupLook()
```

Creates a `HListGroupLook` object. See the class description for details of constructor parameters and default values.

## Methods

### `getElementInsets()`

```
public java.awt.Insets getElementInsets()
```

Retrieve the element insets for this instance of `HListGroupLook`. The element insets control the amount of empty space left between the elements and the border of the `HListGroup` component.

#### Returns:

the element insets, or `null` if insets are not used by this implementation of `HListGroupLook`.

### `getInsets(HVisible)`

```
public java.awt.Insets getInsets(HVisible visible)
```

Determines the insets of this `HLook` , which indicate the size of the border. This area is reserved for the `HLook` to use for drawing borders around the associated `HVisible` .

**Specified By:**

`getInsets(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

the insets of this `HLook` .

**getMaximumSize(HVisible)**

```
public java.awt.Dimension getMaximumSize(HVisible hvisible)
```

Gets the maximum size of the `HVisible` component when drawn with this `HLook` .

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook` ) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If the `HLook` supports the scaling of its content (e.g. an `HGraphicLook` ) then the return value is the current size of the `HVisible` (as returned by `HVisible#getSize` ).
2. If the `HLook` does not support scaling of content or no scaling is requested, and content is set then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If there is no content set then a maximum size of [ `Short.MAX_VALUE` , `Short.MAX_VALUE` ] is returned as a `Dimension`.

**Specified By:**

`getMaximumSize(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating this `HLook` maximum size.

**See Also:**

`getMaximumSize()`

**getMinimumSize(HVisible)**

```
public java.awt.Dimension getMinimumSize(HVisible hvisible)
```

Gets the minimum size of the `HVisible` component when drawn with this `HLook` .

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook` ) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If the `HLook` supports the scaling of its content (e.g. an `HGraphicLook` ) and content is set then the return value is the size of the smallest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.

2. If the `HLook` does not support scaling of content or no scaling is requested, *and* content is set then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If no content is available but a default preferred size has been set using `setDefaultSize(Dimension)` has been called to set then the return value is this value (as obtained with `getDefaultSize()`) plus any additional dimensions that the `HLook` requires for border decoration etc.
4. If there is no content or default size set then the return value is an implementation-specific minimum size plus any additional dimensions that the `HLook` requires for border decoration etc.

**Specified By:**

`getMinimumSize(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating this `HLook` minimum size.

**See Also:**

`getMinimumSize()`

**getNumVisible(HVisible)**

```
public int getNumVisible(HVisible visible)
```

Retrieve the number of visible elements for the specified component.

This method should determine the number of list elements that would be completely visible should the specified component be drawn using this look.

**Parameters:**

`visible` - the `HVisible` to obtain the number of visible elements for.

**Returns:**

the number of visible elements.

**getPreferredSize(HVisible)**

```
public java.awt.Dimension getPreferredSize(HVisible hvisible)
```

Gets the preferred size of the `HVisible` component when drawn with this `HLook`.

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook`) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If a default preferred size has been set for this `HVisible` (using `setDefaultSize(Dimension)`) then the return value is this size (obtained with `getDefaultSize()`) plus any additional dimensions that the `HLook` requires for border decoration etc.
2. If this `HLook` does not support scaling of content or no scaling is requested, and content is present then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If this `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) and content is set then the return value is the current size of the `HVisible` as returned by `getSize()`.
4. If there is no content and no default size set then the return value is the current size of the `HVisible` as



returned by `getSize()`).

**Specified By:**

`getPreferredSize(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating the preferred size of the `HVisible` when drawn with this `HLook`.

**See Also:**

`getPreferredSize()`, `setDefaultSize(Dimension)`

**hitTest(HAdjustmentValue, Point)**

```
public int hitTest(HAdjustmentValue component, java.awt.Point pt)
```

Returns a value which indicates the pointer click position in the on-screen representation of the adjustable component. Note that it is a valid implementation option to always return `ADJUST_NONE`.

**Specified By:**

`hitTest(HAdjustmentValue, Point)` in interface `HAdjustableLook`

**Parameters:**

`component` - the `HAdjustmentValue` component for which the hit position should be calculated.

`pt` - the pointer click point.

**Returns:**

one of `ADJUST_NONE`, `ADJUST_BUTTON_LESS`, `ADJUST_PAGE_LESS`, `ADJUST_THUMB`, `ADJUST_PAGE_MORE` or `ADJUST_BUTTON_MORE`.

**isOpaque(HVisible)**

```
public boolean isOpaque(HVisible visible)
```

Returns true if the entire painted area of the `HVisible` when using this look is fully opaque, i.e. the `showLook(Graphics, HVisible, int)` method guarantees that all pixels are painted in an opaque `Color`.

The default value is implementation specific and depends on the background painting mode of the given `HVisible`. The consequences of an invalid overridden value are implementation specific.

**Specified By:**

`isOpaque(HVisible)` in interface `HLook`

**Parameters:**

`visible` - the visible to test

**Returns:**

true if all the pixels with the `java.awt.Component#getBounds` method of an `HVisible` using this look are fully opaque, i.e. the `showLook(Graphics, HVisible, int)` method guarantees that all pixels are painted in an opaque `Color`, otherwise false.

**showLook(Graphics, HVisible, int)**

```
public void showLook(java.awt.Graphics g, HVisible visible, int state)
```

The `showLook(Graphics, HVisible, int)` method is responsible for repainting the entire `HVisible` component, (including any content set on the component, and the component background), subject to the `clipRect` of the `Graphics` object passed to it.

The `showLook(Graphics, HVisible, int)` method should not modify the `clipRect` of the `Graphics` object that is passed to it.

For looks which draw content (e.g. `HTextLook`, `HGraphicLook` and `HAnimateLook`), if no content is associated with the component, the `showLook(Graphics, HVisible, int)` method paints the component with its current background `Color` according to the `setBackgroundMode(int)` method of `HVisible` and draws any (implementation-specific) borders. Note that by default the background mode is set so as to **not** paint a background. Furthermore on platforms which support transparent colors the background `Color` may be partially or completely transparent.

Any resources **explicitly** associated with an `HLook` should be loaded by the `HLook` during its creation, etc., or via its `setXXX()` methods. Note that the "standard" looks don't load content by default.

This method is called from the `paint(Graphics)` method of `HVisible` and must never be called from elsewhere. Components wishing to redraw themselves should call their `repaint` method in the usual way.

**Specified By:**

`showLook(Graphics, HVisible, int)` in interface `HLook`

**Parameters:**

`g` - the graphics context.

`visible` - the visible.

`state` - the state parameter indicates the state of the visible, allowing the look to render the appropriate content for that state. Note that some components (e.g. `HStaticRange`, `HRange`, `HRangeValue`) do not use state-based content).

**widgetChanged(HVisible, HChangeData[])**

```
public void widgetChanged(HVisible visible, HChangeData[] changes)
```

Called by the `HVisible` whenever its content, state, or any other data changes. See the class description of `HVisible` for more information about the `changes` parameter.

The implementation of this method should work out which graphical areas of the `HVisible` have changed and make any relevant calls to trigger the repainting of those areas.

A minimum implementation of this method could simply call

```
visible.repaint()
```

**Specified By:**

`widgetChanged(HVisible, HChangeData[])` in interface `HLook`

**Parameters:**

`visible` - the `HVisible` which has changed

`changes` - an array containing hint data and associated hint objects. If this argument is `null` a full repaint will be triggered.

# org.havi.ui

# HLook

## Syntax

```
public interface HLook extends java.lang.Cloneable
```

## All Known Subinterfaces:

`HAdjustableLook`

## All Superinterfaces:

`java.lang.Cloneable`

## All Known Implementing Classes:

`HAnimateLook`, `HGraphicLook`, `HTextLook`, `HSinglelineEntryLook`

## Description

The `HLook` interface defines the "look" of a component and may be regarded as a mechanism to allow a "pluggable" paint method to be attached to the component. Instead of having to subclass the entire component to change its look, it is possible to simply implement an `HLook` that will render the component "look" and then associate this `HLook` implementation with the component.

### Borders

An implementation of `HLook` will also include code to draw implementation-specific borders. The application or component can query the reserved space for an `HLook` with the `getInsets(HVisible)` method.

Since the border area is included in the overall size of the component, the border effectively constrains the area available for rendering content to the rectangle which has an upper-left corner location of `(insets.left, insets.top)`, and has a size of width - `(insets.left + insets.right)` by height - `(insets.top + insets.bottom)`.

### Invocation Mechanism

The `showLook(Graphics, HVisible, int)` method of the `HLook` interface will be called by the `havi.ui` framework in response to the paint method of the `HVisible` being called by the AWT lightweight component framework. Applications should simply invoke the component repaint method as in normal AWT, rather than calling the `showLook(Graphics, HVisible, int)` method directly.

The conditions under which the `showLook(Graphics, HVisible, int)` method shall be invoked include the following:

- If the class implements one of the following interfaces, the `showLook(Graphics, HVisible, int)` method shall be invoked when the trigger events specified in the table below are processed:

Interface	Trigger Events
<code>HNavigable</code>	<code>HFocusEvent</code>
<code>HActionable</code>	<code>HFocusEvent</code> , <code>HActionEvent</code>
<code>HSwitchable</code>	<code>HFocusEvent</code> , <code>HActionEvent</code>

HAdjustmentValue HFocusEvent, HAdjustmentEvent

HItemValue HFocusEvent, HItemEvent

HTextView HFocusEvent, HTextEvent

- If the implementing class is an `HVisible`, then the `showLook(Graphics, HVisible, int)` method shall be invoked when content is set on that `HVisible`.
- Content Rendering

Some looks make use of content stored on an associated `HVisible`. These looks are:

- `HAnimateLook` - for animated graphical content
- `HGraphicLook` - for static graphical content
- `HTextLook` - for static textual content
- `HSinglelineEntryLook` - for single line text entry content
- `HMultilineEntryLook` - for multiline text entry content
- `HListGroupLook` - for list element content

Some of these looks may support the scaling and alignment of their content as an implementation option. The scaling and alignment modes supported are specified in the class description of `HVisible`. The table below details which features the content-based platform looks should support:

HLook	Scaling	Alignment
<code>HAnimateLook</code>	Optional	Mandatory
<code>HGraphicLook</code>	Optional	Mandatory
<code>HListGroupLook</code>	Optional	Mandatory
<code>HTextLook</code>	Not used	Mandatory
<code>HSinglelineEntryLook</code>	Not used	Not used
<code>HMultilineEntryLook</code>	Not used	Not used

Where scaling support is optional all implementations must as a minimum support the `RESIZE_NONE` scaling mode. Platforms are *not* required to support scaling of textual content by default.

Looks should use the `getHorizontalAlignment()` and `getVerticalAlignment()` methods to retrieve the current alignment modes, and the `getResizeMode()` method to determine the active scaling mode, where supported. However, note that HAVi platform looks which render text content using the `HDefaultTextLayoutManager` class shall delegate the alignment of text content to the layout manager.

#### Rendering Hints

`HLook` provides a method, `widgetChanged(HVisible, HChangeData[])` which can be called by an `HVisible` with one or more hints to inform the look that something has changed. This method provides information to the look about what has changed, which allows smarter repainting than having the `HVisible` simply call its `repaint` method. The hint constants are defined on `HVisible`. See the class definition for `HVisible` for more information.

#### Private Data

Implementations of `HLook` may store private data on each instance of an `HVisible` to optimize the drawing of that component. However, this is an implementation option. Furthermore such data may be invalidated by another part of the system, for example if `setLook(HLook)` is called on `HVisible`. Therefore if this mechanism is used by implementations of `HLook` those implementations *must* be capable of regenerating such data on the fly, according to the current state of the `HVisible`.

#### Platform Looks

The HAVi UI provides a number of classes implementing the `HLook` interface. Applications wishing to provide their own `HLook` may directly implement this interface or may subclass those provided by the platform.

#### Default Behavior

Unless already specified in a particular `HLook`, implementations of `HLook` should use:

- the foreground color of each associated `HVisible` (using the `java.awt.Component` method `getForeground`) to determine the `Color` to render both the content and border (as necessary)
- the background color of each associated `HVisible` (using the `java.awt.Component` method `getBackground`) to determine the `Color` used to render a rectangular area to fill the component and erase any previous content, as specified by the current background drawing mode for that component - see `setBackgroundMode(int)` on `HVisible`.
- the current alignment mode of each associated `HVisible` (using `getHorizontalAlignment()` and `getVerticalAlignment()`) to determine how to align content.
- the current scaling mode of each associated `HVisible` (using `getResizeMode()`) to determine how to scale content.

#### See Also:

`setLook(HLook)`, `setLookData(Object, Object)`, `paint(Graphics)`, `setBackgroundMode(int)`, `setHorizontalAlignment(int)`, `setVerticalAlignment(int)`, `setResizeMode(int)`, `HTextLook`, `HGraphicLook`, `HAnimateLook`, `HRangeLook`, `HSinglelineEntryLook`, `HMultilineEntryLook`

## Methods

---

### `getInsets(HVisible)`

```
public java.awt.Insets getInsets(HVisible visible)
```

Determines the insets of this `HLook`, which indicate the size of the border. This area is reserved for the `HLook` to use for drawing borders around the associated `HVisible`.

#### Parameters:

`hvisible` - `HVisible` to which this `HLook` is attached.

#### Returns:

the insets of this `HLook`.

---

### `getMaximumSize(HVisible)`

```
public java.awt.Dimension getMaximumSize(HVisible hvisible)
```

Gets the maximum size of the `HVisible` component when drawn with this `HLook`.

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook`) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If the `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) then the return value is the current size of the `HVisible` (as returned by `HVisible#getSize()`).
2. If the `HLook` does not support scaling of content or no scaling is requested, and content is set then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If there is no content set then a maximum size of `[ Short.MAX_VALUE, Short.MAX_VALUE]` is returned as a `Dimension`.

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating this `HLook` maximum size.

**See Also:**

`getMaximumSize()`

---

### **getMinimumSize(HVisible)**

```
public java.awt.Dimension getMinimumSize(HVisible hvisible)
```

Gets the minimum size of the `HVisible` component when drawn with this `HLook`.

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook`) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If the `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) and content is set then the return value is the size of the smallest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
2. If the `HLook` does not support scaling of content or no scaling is requested, *and* content is set then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If no content is available but a default preferred size has been set using `setDefaultSize(Dimension)` has been called to set then the return value is this value (as obtained with `getDefaultSize()`) plus any additional dimensions that the `HLook` requires for border decoration etc.
4. If there is no content or default size set then the return value is an implementation-specific minimum size plus any additional dimensions that the `HLook` requires for border decoration etc.

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating this `HLook` minimum size.

**See Also:**

`getMinimumSize()`

---

### **getPreferredSize(HVisible)**

```
public java.awt.Dimension getPreferredSize(HVisible hvisible)
```

Gets the preferred size of the `HVisible` component when drawn with this `HLook`.

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook`) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If a default preferred size has been set for this `HVisible` (using `setDefaultSize(Dimension)`) then the return value is this size (obtained with `getDefaultSize()`) plus any additional dimensions that the `HLook` requires for border decoration etc.
2. If this `HLook` does not support scaling of content or no scaling is requested, and content is present then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If this `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) and content is set then the return value is the current size of the `HVisible` as returned by `getSize()`.
4. If there is no content and no default size set then the return value is the current size of the `HVisible` as returned by `getSize()`.

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating the preferred size of the `HVisible` when drawn with this `HLook`.

**See Also:**

`getPreferredSize()`, `setDefaultSize(Dimension)`

## **isOpaque(HVisible)**

```
public boolean isOpaque(HVisible visible)
```

Returns true if the entire painted area of the `HVisible` when using this look is fully opaque, i.e. the `showLook(Graphics, HVisible, int)` method guarantees that all pixels are painted in an opaque `Color`.

The default value is implementation specific and depends on the background painting mode of the given `HVisible`. The consequences of an invalid overridden value are implementation specific.

**Parameters:**

`visible` - the visible to test

**Returns:**

true if all the pixels with the `java.awt.Component#getBounds` method of an `HVisible` using this look are fully opaque, i.e. the `showLook(Graphics, HVisible, int)` method guarantees that all pixels are painted in an opaque `Color`, otherwise false.

## **showLook(Graphics, HVisible, int)**

```
public void showLook(java.awt.Graphics g, HVisible visible, int state)
```

The `showLook(Graphics, HVisible, int)` method is responsible for repainting the entire `HVisible` component, (including any content set on the component, and the component background), subject to the `clipRect` of the `Graphics` object passed to it.

The `showLook(Graphics, HVisible, int)` method should not modify the `clipRect` of the `Graphics` object that is passed to it.

For looks which draw content (e.g. `HTextLook`, `HGraphicLook` and `HAnimateLook`), if no content is associated with the component, the `showLook(Graphics, HVisible, int)` method paints the component with its current background `Color` according to the `setBackgroundMode(int)` method of `HVisible` and draws any (implementation-specific) borders. Note that by default the background mode is set so as to **not** paint a background. Furthermore on platforms which support transparent colors the background `Color` may be partially or completely transparent.

Any resources **explicitly** associated with an `HLook` should be loaded by the `HLook` during its creation, etc., or via its `setXXX()` methods. Note that the "standard" looks don't load content by default.

This method is called from the `paint(Graphics)` method of `HVisible` and must never be called from elsewhere. Components wishing to redraw themselves should call their `repaint` method in the usual way.

**Parameters:**

`g` - the graphics context.

`visible` - the visible.

`state` - the state parameter indicates the state of the visible, allowing the look to render the appropriate content for that state. Note that some components (e.g. `HStaticRange`, `HRange`, `HRangeValue`) do not use state-based content).

---

**widgetChanged(HVisible, HChangeData[])**

```
public void widgetChanged(HVisible visible, HChangeData[] changes)
```

Called by the `HVisible` whenever its content, state, or any other data changes. See the class description of `HVisible` for more information about the `changes` parameter.

The implementation of this method should work out which graphical areas of the `HVisible` have changed and make any relevant calls to trigger the repainting of those areas.

A minimum implementation of this method could simply call

```
visible.repaint()
```

**Parameters:**

`visible` - the `HVisible` which has changed

`changes` - an array containing hint data and associated hint objects. If this argument is `null` a full repaint will be triggered.



# org.havi.ui

# HMatte

## Syntax

```
public interface HMatte
```

## All Known Implementing Classes:

```
HImageMatte, HFlatMatte, HFlatEffectMatte, HImageEffectMatte
```

## Description

`HMatte` is the base interface for all matte classes.

Where pixels in a component already have an alpha value (e.g. from an image), the alpha value from the component and the alpha value from the `HMatte` are multiplied together to obtain the actual alpha value to be used for that pixel.

The final displayed value of the component and its `HMatte` is obviously subject to the capabilities of the underlying hardware platform.

# org.havi.ui HMatteException

## Syntax

```
public class HMatteException extends HUIException
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--HUIException
            |
            +--org.havi.ui.HMatteException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

An `HMatteException` is an exception that is thrown when a Component is unable to support the desired `HMatte` effect. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
message	Message explaining why the exception was thrown	null	-	java.lang.Throwable#getMessage

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## See Also:

`HMatteLayer`

## Constructors

### HMatteException()

```
public HMatteException()
```

Creates an `HMatteException` object. See the class description for details of constructor parameters and default values.

---

**HMatteException(String)**

```
public HMatteException(java.lang.String message)
```

Creates an HMatteException object with a specified reason string.

**Parameters:**

message - the reason why the exception was raised

# org.havi.ui HMatteLayer

## Syntax

```
public interface HMatteLayer
```

## All Known Implementing Classes:

`HComponent`, `HContainer`

## Description

This `HMatteLayer` interface enables the presentation of components, together with an associated `HMatte`, for matte compositing.

## See Also:

`HMatte`

## Methods

---

### getMatte()

```
public HMatte getMatte()
```

Get any `HMatte` currently associated with this component.

#### Returns:

the `HMatte` currently associated with this component or null if there is no associated matte.

---

### setMatte(HMatte)

```
public void setMatte(HMatte m)
```

Applies an `HMatte` to this component, for matte compositing. Any existing animated matte must be stopped before this method is called or an `HMatteException` will be thrown.

#### Parameters:

`m` - The `HMatte` to be applied to this component -- note that only one matte may be associated with the component, thus any previous matte will be replaced. If `m` is null, then any matte associated with the component is removed and further calls to `getMatte()` shall return null. The component shall behave as if it had a fully opaque `HFlatMatte` associated with it (i.e an `HFlatMatte` with the default value of 1.0.)

#### Throws:

`HMatteException` - if the `HMatte` cannot be associated with the component. This can occur:

- if the specific matte type is not supported
- if the platform does not support any matte type
- if the component is associated with an already running `HFlatEffectMatte` or `HImageEffectMatte`. The exception is thrown even if `m` is null.

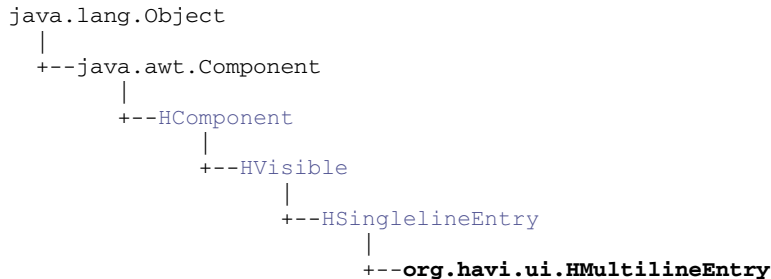
#### See Also:

`HMatte`

# org.havi.ui HMultilineEntry

## Syntax

```
public class HMultilineEntry extends HSinglelineEntry
```



## All Implemented Interfaces:

[HKeyboardInputPreferred](#), [HMatteLayer](#), [HNavigable](#), [HNavigationInputPreferred](#), [HState](#), [HTextValue](#), [java.awt.image.ImageObserver](#), [java.awt.MenuContainer](#), [java.io.Serializable](#), [TestOpacity](#)

## Description

The `HMultilineEntry` is a user interface component used to receive multiple lines of alphanumeric entry from the user. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
width	width of this component in pixels (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
height	height of this component in pixels (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
text	The text within this <code>HMultilineEntry</code> , to be used as the displayed and editable content for all states.	null	<code>setTextContent(String, int)</code>	<code>getTextContent(int)</code>

max-Chars	The maximum number of characters (per line) allowed in this <code>HMultilineEntry</code> .	16 characters	<code>setMaxCharsPerLine(int)</code>	<code>getMaxCharsPerLine()</code>
font	The font to be used for this component.	---	<code>java.awt.Component#setFont.</code>	<code>java.awt.Component#getFont.</code>
color	The color to be used for this component.	---	<code>java.awt.Component#setForeground.</code>	<code>java.awt.Component#getForeground.</code>

#### Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte ( <code>HMatte</code> ).	none (i.e. <code>getMatte()</code> returns null)	<code>setMatte(HMatte)</code>	<code>getMatte()</code>
The text layout manager responsible for text formatting.	An <code>HDefaultTextLayoutManager</code> object.	<code>setTextLayoutManager(HTextLayoutManager)</code>	<code>getTextLayoutManager()</code>
The background painting mode	<code>NO_BACKGROUND_FILL</code>	<code>setBackgroundMode(int)</code>	<code>getBackgroundMode()</code>
The default preferred size	not set (i.e. null) unless specified by width and height parameters	<code>setDefaultSize(Dimension)</code>	<code>getDefaultSize()</code>
The horizontal content alignment	<code>HALIGN_CENTER</code>	<code>setHorizontalAlignment(int)</code>	<code>getHorizontalAlignment()</code>
The vertical content alignment	<code>VALIGN_CENTER</code>	<code>setVerticalAlignment(int)</code>	<code>getVerticalAlignment()</code>
The content scaling mode	<code>RESIZE_NONE</code>	<code>setResizeMode(int)</code>	<code>getResizeMode()</code>
Caret position	At the end of the current text string	<code>setCaretCharPosition(int)</code>	<code>getCaretCharPosition()</code>
Input type	<code>INPUT_ANY</code>	<code>setType(int)</code>	<code>getType()</code>
Customized input range	As for <code>INPUT_ALPHANUMERIC</code>	<code>setValidInput(char[])</code>	<code>getValidInput()</code>
Password protection (the echo character)	Entry is "clear", i.e. not password protected.	<code>setEchoChar(char)</code>	<code>getEchoChar()</code> and <code>echoCharIsSet()</code>
The default "look" for this class.	A platform specific <code>HMultilineEntryLook</code>	<code>setDefaultLook(HMultilineEntryLook)</code>	<code>getDefaultLook()</code>

The "look" for this object.	The <code>HMultilineEntryLook</code> returned from <code>HMultilineEntry.getDefaultLook</code> when this object was created.	<code>setLook(HLook)</code>	<code>getLook()</code>
The gain focus sound.	<code>null</code>	<code>setGainFocusSound(HSound)</code>	<code>getGainFocusSound()</code>
The lose focus sound.	<code>null</code>	<code>setLoseFocusSound(HSound)</code>	<code>getLoseFocusSound()</code>

## Constructors

---

### **HMultilineEntry()**

```
public HMultilineEntry()
```

Creates an `HMultilineEntry` object. See the class description for details of constructor parameters and default values.

---

### **HMultilineEntry(int)**

```
public HMultilineEntry(int maxChars)
```

Creates an `HMultilineEntry` object. See the class description for details of constructor parameters and default values.

---

### **HMultilineEntry(int, int, int, int, int)**

```
public HMultilineEntry(int x, int y, int width, int height, int maxChars)
```

Creates an `HMultilineEntry` object. See the class description for details of constructor parameters and default values.

---

### **HMultilineEntry(String, int, Font, Color)**

```
public HMultilineEntry(java.lang.String text, int maxChars, java.awt.Font font,
    java.awt.Color color)
```

Creates an `HMultilineEntry` object. See the class description for details of constructor parameters and default values.

---

### **HMultilineEntry(String, int, int, int, int, int, Font, Color)**

```
public HMultilineEntry(java.lang.String text, int x, int y, int width, int height,
    int maxChars, java.awt.Font font, java.awt.Color color)
```

Creates an `HMultilineEntry` object. See the class description for details of constructor parameters and default values.

## Methods

---

### caretNextLine()

```
public void caretNextLine()
```

Move the caret to the same column position on the next line. If the caret would be past the end of the text on the line the new caret position will be at the end of the line.

---

### caretPreviousLine()

```
public void caretPreviousLine()
```

Move the caret to the same column position on the previous line. If the caret would be past the end of the text on the line the new caret position will be at the end of the line.

---

### getDefaultLook()

```
public static HSinglelineEntryLook getDefaultLook()
```

Returns the currently set default look for `HMultilineEntry` components.

**Returns:**

The `HMultilineEntryLook` that is used by default when creating a new `HMultilineEntry` component, cast to an `HSinglelineEntryLook`.

---

### setDefaultLook(HMultilineEntryLook)

```
public static void setDefaultLook(HMultilineEntryLook look)
```

Sets the default look for further `HMultilineEntry` Components.

**Parameters:**

`look` - The look that will be used by default when creating a new `HMultilineEntry` component. Note that this parameter may be null, in which case newly created components shall not draw themselves until a non-null look is set using the `setLook(HLook)` method.

---

### setLook(HLook)

```
public void setLook(HLook hlook)
```

Sets the `HLook` for this component.

**Overrides:**

`setLook(HLook)` in class `HSinglelineEntry`

**Parameters:**

`hlook` - The `HLook` that is to be used for this component.

**Throws:**

`HInvalidLookException` - If the Look is not an `HMultilineEntryLook`.



# org.havi.ui HMultilineEntryLook

## Syntax

```
public class HMultilineEntryLook extends HSinglelineEntryLook
```

```
java.lang.Object
|
+--HSinglelineEntryLook
|
+--org.havi.ui.HMultilineEntryLook
```

## All Implemented Interfaces:

java.lang.Cloneable, HLook

## Description

The `HMultilineEntryLook` class is used by the `HMultilineEntry` component to display the entering of text. This look will be provided by the platform and the exact way in which it is rendered will be platform dependent.

The `HMultilineEntryLook` class draws the content set on an `HMultilineEntry`. It uses the `getTextContent(int)` method to determine the content to render. The interaction state of the `HMultilineEntry` is ignored.

This is the default look that is used by `HMultilineEntry` and its subclasses. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## See Also:

`HMultilineEntry`

## Constructors

### HMultilineEntryLook()

```
public HMultilineEntryLook()
```

Creates a `HMultilineEntryLook` object. See the class description for details of constructor parameters and default values.

## Methods

---

### **getCaretCharPositionForLine(HVisible, int)**

```
public int getCaretCharPositionForLine(HVisible visible, int line)
```

Returns the character position of the caret within the content string if it were to be moved vertically to the given 'line'. A line is identified by its first character as obtained from the `getSoftLineBreakPositions()` method. If an invalid line is specified an `IllegalArgumentException` is thrown. If it cannot be moved the nearest position should be returned.

**Parameters:**

`visible` - a multiline text entry component.

Note that if this component is not actually using this `HLook` to display itself the return value of this method will probably be wrong. Application authors should take care to only call this method with components which are using this instance of the `HMultilineEntryLook`.

`line` - the line number for which the caret position is requested

**Returns:**

the caret position (as defined in `getCaretCharPosition()`) to move the caret to if it were moved to this line.

**See Also:**

`getCaretCharPosition()`, `getSoftLineBreakPositions(HVisible)`

---

### **getCaretPositionNextLine(HVisible)**

```
public int getCaretPositionNextLine(HVisible visible)
```

Returns the character position of the caret within the content string if it were to be moved down one line.

**Parameters:**

`visible` - a multiline text entry component.

Note that if this component is not actually using this `HLook` to display itself the return value of this method will probably be wrong. Application authors should take care to only call this method with components which are using this instance of the `HMultilineEntryLook`.

**Returns:**

the caret position (as defined in `getCaretCharPosition()`) to move the caret to if it were moved down one line, or the nearest position if this is not possible.

---

### **getCaretPositionPreviousLine(HVisible)**

```
public int getCaretPositionPreviousLine(HVisible visible)
```

Returns the character position of the caret within the content string if it were to be moved up one line.

**Parameters:**

`visible` - a multiline text entry component.

Note that if this component is not actually using this `HLook` to display itself the return value of this method will probably be wrong. Application authors should take care to only call this method with components which are using this instance of the `HMultilineEntryLook`.

**Returns:**

the caret position (as defined in `getCaretCharPosition()`) to move the caret to if it were moved up one line, or the nearest position if this is not possible.

---

**getInsets(HVisible)**

```
public java.awt.Insets getInsets(HVisible visible)
```

Determines the insets of this `HLook` , which indicate the size of the border. This area is reserved for the `HLook` to use for drawing borders around the associated `HVisible` .

**Overrides:**

`getInsets(HVisible)` in class `HSinglelineEntryLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

the insets of this `HLook` .

---

**getMaximumSize(HVisible)**

```
public java.awt.Dimension getMaximumSize(HVisible hvisible)
```

Gets the maximum size of the `HVisible` component when drawn with this `HLook` .

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook` ) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If the `HLook` supports the scaling of its content (e.g. an `HGraphicLook` ) then the return value is the current size of the `HVisible` (as returned by `HVisible#getSize` ).
2. If the `HLook` does not support scaling of content or no scaling is requested, and content is set then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If there is no content set then a maximum size of `[ Short.MAX_VALUE, Short.MAX_VALUE]` is returned as a `Dimension`.

**Overrides:**

`getMaximumSize(HVisible)` in class `HSinglelineEntryLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating this `HLook` maximum size.

**See Also:**

`getMaximumSize()`

---

**getMinimumSize(HVisible)**

```
public java.awt.Dimension getMinimumSize(HVisible hvisible)
```

Gets the minimum size of the `HVisible` component when drawn with this `HLook` .

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook` ) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If the `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) and content is set then the return value is the size of the smallest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
2. If the `HLook` does not support scaling of content or no scaling is requested, *and* content is set then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If no content is available but a default preferred size has been set using `setDefaultSize(Dimension)` has been called to set then the return value is this value (as obtained with `getDefaultSize()`) plus any additional dimensions that the `HLook` requires for border decoration etc.
4. If there is no content or default size set then the return value is an implementation-specific minimum size plus any additional dimensions that the `HLook` requires for border decoration etc.

**Overrides:**

`getMinimumSize(HVisible)` in class `HSinglelineEntryLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating this `HLook` minimum size.

**See Also:**

`getMinimumSize()`

**getPreferredSize(HVisible)**

```
public java.awt.Dimension getPreferredSize(HVisible hvisible)
```

Gets the preferred size of the `HVisible` component when drawn with this `HLook`.

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook`) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If a default preferred size has been set for this `HVisible` (using `setDefaultSize(Dimension)`) then the return value is this size (obtained with `getDefaultSize()`) plus any additional dimensions that the `HLook` requires for border decoration etc.
2. If this `HLook` does not support scaling of content or no scaling is requested, and content is present then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If this `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) and content is set then the return value is the current size of the `HVisible` as returned by `getSize()`.
4. If there is no content and no default size set then the return value is the current size of the `HVisible` as returned by `getSize()`.

**Overrides:**

`getPreferredSize(HVisible)` in class `HSinglelineEntryLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating the preferred size of the `HVisible` when drawn with this `HLook`.

**See Also:**

```
getPreferredSize(), setDefaultSize(Dimension)
```

---

### getSoftLineBreakPositions(HVisible)

```
public int[] getSoftLineBreakPositions(HVisible visible)
```

Returns the positions within the content string of all those characters that start on a new 'line' (including those following \n), in order from the first line to the last, including the line starting at 0.

**Parameters:**

`visible` - a multiline text entry component.

Note that if this component is not actually using this `HLook` to display itself the return value of this method will probably be wrong. Application authors should take care to only call this method with components which are using this instance of the `HMultilineEntryLook`.

**Returns:**

the positions of 'soft line breaks' introduced by the `HLook`.

---

### getVisibleSoftLineBreakPositions(HVisible)

```
public int[] getVisibleSoftLineBreakPositions(HVisible visible)
```

Returns the starting positions of lines currently shown within the `HVisible`. Lines are identified by the positions of those characters that start on a new 'line' (including those following \n), in order, from the first visible line to the last. This method can be used to calculate the number of visible lines currently shown in the look as well as the line shown at the top of the visible window.

**Parameters:**

`visible` - a multiline text entry component.

**Returns:**

the starting positions of the lines currently shown within the `HVisible`.

---

### isOpaque(HVisible)

```
public boolean isOpaque(HVisible visible)
```

Returns true if the entire painted area of the `HVisible` when using this look is fully opaque, i.e. the `showLook(Graphics, HVisible, int)` method guarantees that all pixels are painted in an opaque `Color`.

The default value is implementation specific and depends on the background painting mode of the given `HVisible`. The consequences of an invalid overridden value are implementation specific.

**Overrides:**

`isOpaque(HVisible)` in class `HSinglelineEntryLook`

**Parameters:**

`visible` - the visible to test

**Returns:**

true if all the pixels with the `java.awt.Component#getBounds` method of an `HVisible` using this look are fully opaque, i.e. the `showLook(Graphics, HVisible, int)` method guarantees that all pixels are painted in an opaque `Color`, otherwise false.

---

### showLook(Graphics, HVisible, int)

```
public void showLook(java.awt.Graphics g, HVisible visible, int state)
```

The `showLook(Graphics, HVisible, int)` method is responsible for repainting the entire `HVisible` component, (including any content set on the component, and the component background), subject to the `clipRect` of the `Graphics` object passed to it.

The `showLook(Graphics, HVisible, int)` method should not modify the `clipRect` of the `Graphics` object that is passed to it.

For looks which draw content (e.g. `HTextLook`, `HGraphicLook` and `HAnimateLook`), if no content is associated with the component, the `showLook(Graphics, HVisible, int)` method paints the component with its current background `Color` according to the `setBackgroundMode(int)` method of `HVisible` and draws any (implementation-specific) borders. Note that by default the background mode is set so as to **not** paint a background. Furthermore on platforms which support transparent colors the background `Color` may be partially or completely transparent.

Any resources **explicitly** associated with an `HLook` should be loaded by the `HLook` during its creation, etc., or via its `setXXX()` methods. Note that the "standard" looks don't load content by default.

This method is called from the `paint(Graphics)` method of `HVisible` and must never be called from elsewhere. Components wishing to redraw themselves should call their repaint method in the usual way.

**Overrides:**

`showLook(Graphics, HVisible, int)` in class `HSinglelineEntryLook`

**Parameters:**

`g` - the graphics context.

`visible` - the visible.

`state` - the state parameter indicates the state of the visible, allowing the look to render the appropriate content for that state. Note that some components (e.g. `HStaticRange`, `HRange`, `HRangeValue`) do not use state-based content).

**widgetChanged(HVisible, HChangeData[])**

```
public void widgetChanged(HVisible visible, HChangeData[] changes)
```

Called by the `HVisible` whenever its content, state, or any other data changes. See the class description of `HVisible` for more information about the `changes` parameter.

The implementation of this method should work out which graphical areas of the `HVisible` have changed and make any relevant calls to trigger the repainting of those areas.

A minimum implementation of this method could simply call

```
visible.repaint()
```

**Overrides:**

`widgetChanged(HVisible, HChangeData[])` in class `HSinglelineEntryLook`

**Parameters:**

`visible` - the `HVisible` which has changed

`changes` - an array containing hint data and associated hint objects. If this argument is `null` a full repaint will be triggered.

# org.havi.ui

## HNavigable

### Syntax

`public interface HNavigable extends HNavigationInputPreferred`

### All Known Subinterfaces:

`HActionable, HAdjustmentValue, HItemValue, HSwitchable, HTextValue`

### All Superinterfaces:

`HNavigationInputPreferred`

### All Known Implementing Classes:

`HIcon, HRange, HText, HAnimation`

### Description

This interface is implemented by all HAVi UI components that can be navigated to by the user (i.e. components which can gain focus).

#### Event Behavior

Subclasses of `HComponent` which implement `HNavigable` must respond to `HFocusEvent` events.

Applications should assume that classes which implement `HNavigable` can only generate events of the type `HFocusEvent` in response to other types of input event.

An application may add one or more `HFocusListener` listeners to the component. The `focusGained` and `focusLost` methods of the `HFocusListener` are invoked whenever the `HNavigable` gains or loses focus.

An `HNavigable` has an arbitrary focus traversal table associated with it (see `setMove(int, HNavigable)` and `getMove(int)`). This mechanism allows the four-way focus behavior of a set of components to be set (see `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)`, `setMove(int, HNavigable)` and `getMove(int)`).

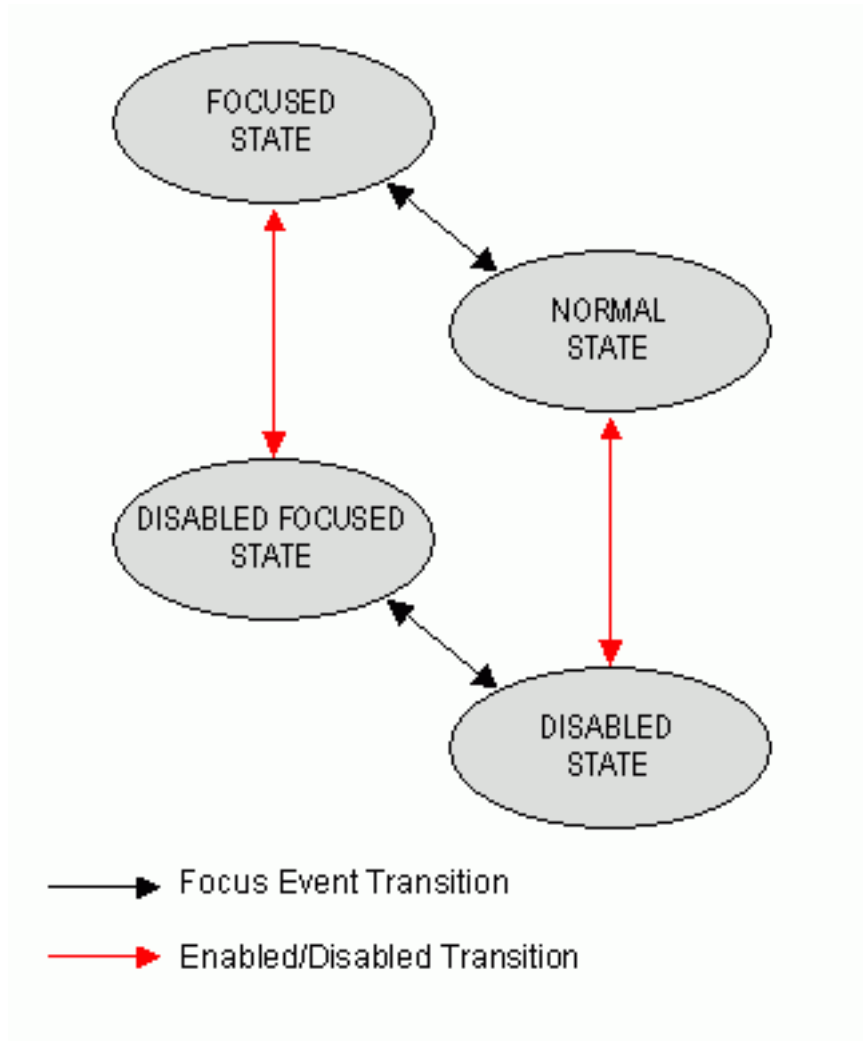
HAVi focus events are discussed in detail in the `HNavigationInputPreferred` interface description.

#### Interaction States

The following interaction states are valid for this `HNavigable` component:

- `NORMAL_STATE`
- `FOCUSED_STATE`
- `DISABLED_STATE`
- `DISABLED_FOCUSED_STATE`

The state machine diagram below shows the valid state transitions for an `HNavigable` component.



#### Platform Classes

The following HAVi platform classes implement or inherit the `HNavigable` interface. These classes shall all generate `HFocusEvent` events in addition to any other events specified in the respective class descriptions.

- `HAnimation`
- `HIcon`
- `HText`
- `HRange`
- `HGraphicButton`
- `HTextButton`
- `HToggleButton`
- `HListGroup`
- `HSinglelineEntry`
- `HMultilineEntry`
- `HRangeValue`

#### See Also:

`HNavigationInputPreferred`, `HFocusEvent`, `HFocusListener`



## Methods

---

### getGainFocusSound()

```
public HSound getGainFocusSound()
```

Get the sound associated with the gain focus event.

**Returns:**

The sound played when the component gains focus. If no sound is associated with gaining focus, then null shall be returned.

---

### getLoseFocusSound()

```
public HSound getLoseFocusSound()
```

Get the sound associated with the lost focus event.

**Returns:**

The sound played when the component loses focus. If no sound is associated with losing focus, then null shall be returned.

---

### getMove(int)

```
public HNavigable getMove(int keyCode)
```

Provides the `HNavigable` object that is navigated to when a particular key is pressed.

**Parameters:**

`keyCode` - The key code of the pressed key.

**Returns:**

Returns the `HNavigable` object, or if no `HNavigable` is associated with the `keyCode` then returns null.

---

### isSelected()

```
public boolean isSelected()
```

Indicates if this component has focus.

**Returns:**

`true` if the component has focus, otherwise returns `false`.

---

### setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
                             HNavigable right)
```

Set the focus control for an `HNavigable` component. Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is a convenience function for application programmers where a standard up, down, left and right focus traversal between components is required.

Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

Note that this API does not prevent the creation of "isolated" `HNavigable` components --- authors should endeavor to avoid confusing the user.

**Parameters:**

`up` - The `HNavigable` component to move to, when the user generates a `VK_UP` `KeyEvent`. If there is no `HNavigable` component to move "up" to, then null should be specified.

`down` - The `HNavigable` component to move to, when the user generates a `VK_DOWN` `KeyEvent`. If there is no `HNavigable` component to move "down" to, then null should be specified.

`left` - The `HNavigable` component to move to, when the user generates a `VK_LEFT` `KeyEvent`. If there is no `HNavigable` component to move "left" to, then null should be specified.

`right` - The `HNavigable` component to move to, when the user generates a `VK_RIGHT` `KeyEvent`. If there is no `HNavigable` component to move "right" to, then null should be specified.

**setGainFocusSound(HSound)**

```
public void setGainFocusSound(HSound sound)
```

Associate a sound with gaining focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_GAINED`. This sound will start to be played when an object implementing this interface gains focus. It is not guaranteed to be played to completion. If the object implementing this interface loses focus before the audio completes playing, the audio will be truncated. Applications wishing to ensure the audio is always played to completion must implement special logic to slow down the focus transitions.

By default, an `HNavigable` object does not have any gain focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

**Parameters:**

`sound` - the sound to be played, when the component gains focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

**setLoseFocusSound(HSound)**

```
public void setLoseFocusSound(HSound sound)
```

Associate a sound with losing focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_LOST`. This sound will start to be played when an object implementing this interface loses focus. It is not guaranteed to be played to completion. It is implementation dependent whether and when this sound will be truncated by any gain focus sound played by the next object to gain focus.

By default, an `HNavigable` object does not have any lose focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

**Parameters:**

`sound` - the sound to be played, when the component loses focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

**setMove(int, HNavigable)**

```
public void setMove(int keyCode, HNavigable target)
```

Defines the navigation path from the current `HNavigable` to another `HNavigable` when a particular key is pressed.

Note that `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

**Parameters:**

`keyCode` - The key code of the pressed key. Any numerical keycode is allowed, but the platform may not be able to generate all keycodes. Application authors should only use keys for which `HRcCapabilities.isSupported()` or `HKeyCapabilities.isSupported()` returns true.

`target` - The target `HNavigable` object that should be navigated to. If a target is to be removed from a particular navigation path, then `null` should be specified.

# org.havi.ui

# HNavigationInputPreferred

## Syntax

```
public interface HNavigationInputPreferred
```

## All Known Subinterfaces:

```
HActionable, HAdjustmentValue, HItemValue, HNavigable, HSwitchable, HTextValue
```

## Description

A component which implements `HNavigationInputPreferred` indicates that this component expects to receive `HFocusEvent` events. The focus event system in HAVi is designed to be compatible with standard AWT focus mechanisms while supporting key event-based focus traversal for HAVi UI components.

All interoperable implementations of the `HNavigationInputPreferred` interface must extend `HComponent`.

Components which implement `HNavigationInputPreferred` to handle `HFocusEvent` events can optionally manage focus traversal based on keyboard input events, in addition to the normal semantics of the `FOCUS_GAINED` and `FOCUS_LOST` event types. The focus traversal mechanism specified by the HAVi UI `HNavigable` interface is one such system.

In the case where such an implementation requires specific keys to manage focus traversal the `getNavigationKeys()` method is provided to allow the HAVi platform to query the set of keys for which a navigation target has been set. When such a component has the input focus, platforms without a physical means of generating the desired keystrokes shall provide another means for navigation e.g. by offering an on-screen "virtual" keyboard. Applications can query the system about the support of specific keyCodes through the `isSupported(int)` method.

The keyCodes for navigation keystrokes generated on the `HNavigationInputPreferred` will be passed to the `HNavigationInputPreferred` as an `HFocusEvent` transferId through the `processHFocusEvent(HFocusEvent)` method. No `HKeyEvent` will be generated on the `HNavigationInputPreferred` as a result of these keystrokes.

Note that the `java.awt.Component` method `isFocusTraversable` should always return true for a `java.awt.Component` implementing this interface.

## Methods

---

### `getNavigationKeys()`

```
public int[] getNavigationKeys()
```

Retrieve the set of key codes which this component maps to navigation targets.

#### Returns:

an array of key codes, or `null` if no navigation targets are set on this component.

---

**processHFocusEvent(HFocusEvent)**

```
public void processHFocusEvent(HFocusEvent evt)
```

Process an `HFocusEvent` sent to this `HNavigationInputPreferred`.

**Parameters:**

`evt` - the `HFocusEvent` to process.

# org.havi.ui

# HNoInputPreferred

## Syntax

```
public interface HNoInputPreferred
```

## All Known Implementing Classes:

```
HStaticIcon, HStaticRange, HStaticText, HStaticAnimation
```

## Description

A component which implements `HNoInputPreferred` indicates that the user cannot navigate to this component.

All interoperable implementations of the `HNoInputPreferred` interface must extend `HComponent` .

In some cases a component which implements this interface will be extended, and that component will implement another "XxxInputPreferred" interface. In all cases, these other interfaces take precedence.

# org.havi.ui

# HOrientable

## Syntax

```
public interface HOrientable
```

## All Known Subinterfaces:

```
HAdjustmentInputPreferred, HAdjustmentValue, HItemValue,  
HSelectionInputPreferred
```

## All Known Implementing Classes:

```
HStaticRange
```

## Description

The `HOrientable` interface provides support for components which have an orientation.

All interoperable implementations of the `HOrientable` interface must extend `HComponent`.

## Fields

---

### ORIENT\_BOTTOM\_TO\_TOP

```
public static final int ORIENT_BOTTOM_TO_TOP
```

A constant which specifies that the `HOrientable` should be rendered with a vertical orientation, with the minimum value on the bottom, and the maximum value on the top.

---

### ORIENT\_LEFT\_TO\_RIGHT

```
public static final int ORIENT_LEFT_TO_RIGHT
```

A constant which specifies that the `HOrientable` should be rendered with a horizontal orientation, with the minimum value on the left side, and the maximum value on the right side.

---

### ORIENT\_RIGHT\_TO\_LEFT

```
public static final int ORIENT_RIGHT_TO_LEFT
```

A constant which specifies that the `HOrientable` should be rendered with a horizontal orientation, with the minimum value on the right side, and the maximum value on the left side.

---

### ORIENT\_TOP\_TO\_BOTTOM

```
public static final int ORIENT_TOP_TO_BOTTOM
```

A constant which specifies that the `HOrientable` should be rendered with a vertical orientation, with the minimum value on the top, and the maximum of the range on the bottom.

## Methods

---

### **getOrientation()**

```
public int getOrientation()
```

Retrieve the orientation of the `HOrientable`. The orientation controls how an associated `HLook` lays out the component and affects the visual behavior of the `HAdjustmentEvent` and `HItemEvent` events. For example, the system might use this information to select appropriate key mappings for these events.

**Returns:**

one of `ORIENT_LEFT_TO_RIGHT`, `ORIENT_RIGHT_TO_LEFT`, `ORIENT_TOP_TO_BOTTOM`, or `ORIENT_BOTTOM_TO_TOP`.

---

### **setOrientation(int)**

```
public void setOrientation(int orient)
```

Set the orientation of the `HOrientable`. The orientation controls how the associated `HLook` lays out the component.

**Parameters:**

`orient` - one of `ORIENT_LEFT_TO_RIGHT`, `ORIENT_RIGHT_TO_LEFT`, `ORIENT_TOP_TO_BOTTOM`, or `ORIENT_BOTTOM_TO_TOP`.



# org.havi.ui HPermissionDeniedException

## Syntax

```
public class HPermissionDeniedException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
|
+--java.lang.Exception
|
+--org.havi.ui.HPermissionDeniedException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Thrown when an application attempts to control a device or feature it does not have the right to do at the time of making the call. The precise conditions of this are defined in more detail in the places where this exception is thrown. It could be when the application does not have some underlying resource reserved.

This exception is not related in any way to the Java security model or to `java.lang.SecurityException`. Methods where an application could be denied the right to control a device or feature by security policy also must throw a `java.lang.SecurityException`. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
message	Message explaining why the exception was thrown	null	-	java.lang.Throwable#getMessage

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Constructors

### HPermissionDeniedException()

```
public HPermissionDeniedException()
```

Creates an `HPermissionDeniedException` object. See the class description for details of constructor parameters and default values.

---

**HPermissionDeniedException(String)**

```
public HPermissionDeniedException(java.lang.String message)
```

Creates an HPermissionDeniedException object with a specified reason string.

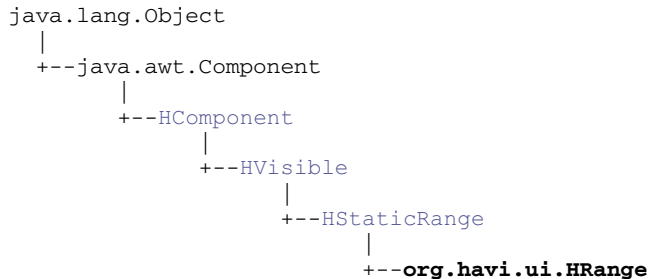
**Parameters:**

message - the reason why the exception was raised

# org.havi.ui HRange

## Syntax

```
public class HRange extends HStaticRange implements HNavigable
```



## Direct Known Subclasses:

[HRangeValue](#)

## All Implemented Interfaces:

[HMatteLayer](#), [HNavigable](#), [HNavigationInputPreferred](#), [HNoInputPreferred](#), [HOrientable](#), [HState](#), [java.awt.image.ImageObserver](#), [java.awt.MenuContainer](#), [java.io.Serializable](#), [TestOpacity](#)

## Description

The [HRange](#) is a user interface component used to display a value within a fixed range (as [org.havi.ui.HStaticRange](#) [HStaticRange](#)) which also enables a user to navigate to it, i.e. it can have the input focus. By default it uses the [HRangeLook](#) to render itself. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	<a href="#">java.awt.Component#setBounds</a>	<a href="#">java.awt.Component#getBounds</a>
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	<a href="#">java.awt.Component#setBounds</a>	<a href="#">java.awt.Component#getBounds</a>
width	width of this component in pixels (subject to layout management).	---	<a href="#">java.awt.Component#setBounds</a>	<a href="#">java.awt.Component#getBounds</a>
height	height of this component in pixels (subject to layout management).	---	<a href="#">java.awt.Component#setBounds</a>	<a href="#">java.awt.Component#getBounds</a>

orientation	The "orientation" of the range object.	ORIENT_LEFT_T O_RIGHT	setOrientation( int)	getOrientation( )
minimum	The minimum value that can be returned by this range object.	0	setRange(int, int)	getMinValue()
maximum	The maximum value that can be returned by this range object.	100	setRange(int, int)	getMaxValue()
value	The current value returned by this range object.	0	setValue(int)	getValue()

#### Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte ( <code>HMatte</code> ).	none (i.e. <code>getMatte()</code> returns null)	<code>setMatte(HMatte)</code>	<code>getMatte()</code>
The text layout manager responsible for text formatting.	An <code>HDefaultTextLayoutManager</code> object.	<code>setTextLayoutManager(HTextLayoutManager)</code>	<code>getTextLayoutManager()</code>
The background painting mode	<code>NO_BACKGROUND_FILL</code>	<code>setBackgroundMode(int)</code>	<code>getBackgroundMode()</code>
The default preferred size	not set (i.e. null) unless specified by width and height parameters	<code>setDefaultSize(Dimension)</code>	<code>getDefaultSize()</code>
The horizontal content alignment	<code>HALIGN_CENTER</code>	<code>setHorizontalAlignment(int)</code>	<code>getHorizontalAlignment()</code>
The vertical content alignment	<code>VALIGN_CENTER</code>	<code>setVerticalAlignment(int)</code>	<code>getVerticalAlignment()</code>
The content scaling mode	<code>RESIZE_NONE</code>	<code>setResizeMode(int)</code>	<code>getResizeMode()</code>
The default "look" for this class.	A platform specific <code>HRangeLook</code>	<code>setDefaultLook(HRangeLook)</code>	<code>getDefaultLook()</code>
The "look" for this object.	The <code>HRangeLook</code> returned from <code>HRange.getDefaultLook</code> when this object was created.	<code>setLook(HLook)</code>	<code>getLook()</code>

The offsets for the "thumb" of this range control	min = 0, max = 0	setThumbOffsets(int, int)	getThumbMinOffset() / getThumbMaxOffset()
The behavior of this range object with respect to its "thumb" values	SLIDER_BEHAVIOR	setBehavior(int)	getBehavior()
The gain focus sound.	null	setGainFocusSound(HSound)	getGainFocusSound()
The lose focus sound.	null	setLoseFocusSound(HSound)	getLoseFocusSound()

**See Also:**

[HStaticRange](#), [HNavigable](#)

## Constructors

---

**HRange()**

```
public HRange()
```

Creates an [HRange](#) object. See the class description for details of constructor parameters and default values.

---

**HRange(int, int, int, int)**

```
public HRange(int orientation, int minimum, int maximum, int value)
```

Creates an [HRange](#) object. See the class description for details of constructor parameters and default values.

---

**HRange(int, int, int, int, int, int, int, int)**

```
public HRange(int orientation, int minimum, int maximum, int value, int x, int y,
              int width, int height)
```

Creates an [HRange](#) object. See the class description for details of constructor parameters and default values.

## Methods

---

**getDefaultLook()**

```
public static HRangeLook getDefaultLook()
```

Returns the currently set default [HLook](#) for [HRange](#) components.

**Returns:**

The `HLook` that is used by default when creating a new `HRange` component.

---

**getGainFocusSound()**

```
public HSound getGainFocusSound()
```

Get the sound associated with the gain focus event.

**Specified By:**

```
getGainFocusSound() in interface HNavigable
```

**Returns:**

The sound played when the component gains focus. If no sound is associated with gaining focus, then null shall be returned.

---

**getLoseFocusSound()**

```
public HSound getLoseFocusSound()
```

Get the sound associated with the lost focus event.

**Specified By:**

```
getLoseFocusSound() in interface HNavigable
```

**Returns:**

The sound played when the component loses focus. If no sound is associated with losing focus, then null shall be returned.

---

**getMove(int)**

```
public HNavigable getMove(int keyCode)
```

Provides the `HNavigable` object that is navigated to when a particular key is pressed.

**Specified By:**

```
getMove(int) in interface HNavigable
```

**Parameters:**

`keyCode` - The key code of the pressed key.

**Returns:**

Returns the `HNavigable` object, or if no `HNavigable` is associated with the `keyCode` then returns null.

---

**getNavigationKeys()**

```
public int[] getNavigationKeys()
```

Retrieve the set of key codes which this component maps to navigation targets.

**Specified By:**

```
getNavigationKeys() in interface HNavigationInputPreferred
```

**Returns:**

an array of key codes, or null if no navigation targets are set on this component.

---

**isSelected()**

```
public boolean isSelected()
```

Indicates if this component has focus.

**Specified By:**

`isSelected()` in interface `HNavigable`

**Returns:**

`true` if the component has focus, otherwise returns `false`.

**processHFocusEvent(HFocusEvent)**

```
public void processHFocusEvent(HFocusEvent evt)
```

Process an `HFocusEvent` sent to this `HRange`.

**Specified By:**

`processHFocusEvent(HFocusEvent)` in interface `HNavigationInputPreferred`

**Parameters:**

`evt` - the `HFocusEvent` to process.

**setDefaultLook(HRangeLook)**

```
public static void setDefaultLook(HRangeLook look)
```

Sets the default `HLook` for further `HRange` Components.

**Parameters:**

`look` - The `HLook` that will be used by default when creating a new `HRange` component. Note that this parameter may be null, in which case newly created components shall not draw themselves until a non-null look is set using the `setLook(HLook)` method.

**setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)**

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
    HNavigable right)
```

Set the focus control for an `HNavigable` component. Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is a convenience function for application programmers where a standard up, down, left and right focus traversal between components is required.

Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

Note that this API does not prevent the creation of "isolated" `HNavigable` components --- authors should endeavor to avoid confusing the user.

**Specified By:**

`setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` in interface `HNavigable`

**Parameters:**

`up` - The `HNavigable` component to move to, when the user generates a `VK_UP` KeyEvent. If there is no `HNavigable` component to move "up" to, then null should be specified.

`down` - The `HNavigable` component to move to, when the user generates a `VK_DOWN` KeyEvent. If there is no `HNavigable` component to move "down" to, then null should be specified.

`left` - The `HNavigable` component to move to, when the user generates a `VK_LEFT` KeyEvent. If there is no `HNavigable` component to move "left" to, then null should be specified.

`right` - The `HNavigable` component to move to, when the user generates a `VK_RIGHT` `KeyEvent`. If there is no `HNavigable` component to move "right" to, then null should be specified.

---

### setGainFocusSound(HSound)

```
public void setGainFocusSound(HSound sound)
```

Associate a sound with gaining focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_GAINED`. This sound will start to be played when an object implementing this interface gains focus. It is not guaranteed to be played to completion. If the object implementing this interface loses focus before the audio completes playing, the audio will be truncated. Applications wishing to ensure the audio is always played to completion must implement special logic to slow down the focus transitions.

By default, an `HNavigable` object does not have any gain focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

#### Specified By:

`setGainFocusSound(HSound)` in interface `HNavigable`

#### Parameters:

`sound` - the sound to be played, when the component gains focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

---

### setLoseFocusSound(HSound)

```
public void setLoseFocusSound(HSound sound)
```

Associate a sound with losing focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_LOST`. This sound will start to be played when an object implementing this interface loses focus. It is not guaranteed to be played to completion. It is implementation dependent whether and when this sound will be truncated by any gain focus sound played by the next object to gain focus.

By default, an `HNavigable` object does not have any lose focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

#### Specified By:

`setLoseFocusSound(HSound)` in interface `HNavigable`

#### Parameters:

`sound` - the sound to be played, when the component loses focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

---

### setMove(int, HNavigable)

```
public void setMove(int keyCode, HNavigable target)
```

Defines the navigation path from the current `HNavigable` to another `HNavigable` when a particular key is pressed.

Note that `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

#### Specified By:

`setMove(int, HNavigable)` in interface `HNavigable`



**Parameters:**

`keyCode` - The key code of the pressed key. Any numerical keycode is allowed, but the platform may not be able to generate all keycodes. Application authors should only use keys for which `HRcCapabilities.isSupported()` or `HKeyCapabilities.isSupported()` returns true.

`target` - The target `HNavigable` object that should be navigated to. If a target is to be removed from a particular navigation path, then `null` should be specified.

# org.havi.ui

# HRangeLook

## Syntax

public class HRangeLook implements HAdjustableLook

```
java.lang.Object
|
+--org.havi.ui.HRangeLook
```

## All Implemented Interfaces:

java.lang.Cloneable, HAdjustableLook, HLook

## Description

The `HRangeLook` class is used to display a slider or scrollbar type range control on screen. It is used by specific `HVisible` components which implement the `HAdjustmentValue` interface to present a graphical representation of the value to the user. This look will be provided by the platform and the exact way in which it is rendered will be platform dependant.

This look does NOT use content set on the `HVisible`.

This is the default look that is used by `HStaticRange` and its subclasses. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
-----------	-------------	---------------	------------	------------

None.

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

None.

## See Also:

[HRange](#)

## Constructors

### HRangeLook()

```
public HRangeLook()
```

Creates an `HRangeLook` object. See the class description for details of constructor parameters and default values.

## Methods

---

### getInsets(HVisible)

```
public java.awt.Insets getInsets(HVisible visible)
```

Determines the insets of this `HLook` , which indicate the size of the border. This area is reserved for the `HLook` to use for drawing borders around the associated `HVisible` .

**Specified By:**

`getInsets(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

the insets of this `HLook` .

---

### getMaximumSize(HVisible)

```
public java.awt.Dimension getMaximumSize(HVisible hvisible)
```

Gets the maximum size of the `HVisible` component when drawn with this `HLook` .

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook` ) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If the `HLook` supports the scaling of its content (e.g. an `HGraphicLook` ) then the return value is the current size of the `HVisible` (as returned by `HVisible#getSize` ).
2. If the `HLook` does not support scaling of content or no scaling is requested, and content is set then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If there is no content set then a maximum size of [ `Short.MAX_VALUE` , `Short.MAX_VALUE` ] is returned as a `Dimension`.

**Specified By:**

`getMaximumSize(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating this `HLook` maximum size.

**See Also:**

`getMaximumSize()`

---

### getMinimumSize(HVisible)

```
public java.awt.Dimension getMinimumSize(HVisible hvisible)
```

Gets the minimum size of the `HVisible` component when drawn with this `HLook` .

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this

algorithm `HLook` classes that do not use content (e.g. `HRangeLook`) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If the `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) and content is set then the return value is the size of the smallest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
2. If the `HLook` does not support scaling of content or no scaling is requested, *and* content is set then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If no content is available but a default preferred size has been set using `setDefaultSize(Dimension)` has been called to set then the return value is this value (as obtained with `getDefaultSize()`) plus any additional dimensions that the `HLook` requires for border decoration etc.
4. If there is no content or default size set then the return value is an implementation-specific minimum size plus any additional dimensions that the `HLook` requires for border decoration etc.

**Specified By:**

`getMinimumSize(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating this `HLook` minimum size.

**See Also:**

`getMinimumSize()`

### **getPreferredSize(HVisible)**

```
public java.awt.Dimension getPreferredSize(HVisible hvisible)
```

Gets the preferred size of the `HVisible` component when drawn with this `HLook`.

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook`) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If a default preferred size has been set for this `HVisible` (using `setDefaultSize(Dimension)`) then the return value is this size (obtained with `getDefaultSize()`) plus any additional dimensions that the `HLook` requires for border decoration etc.
2. If this `HLook` does not support scaling of content or no scaling is requested, and content is present then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If this `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) and content is set then the return value is the current size of the `HVisible` as returned by `getSize()`.
4. If there is no content and no default size set then the return value is the current size of the `HVisible` as returned by `getSize()`.

**Specified By:**

`getPreferredSize(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating the preferred size of the `HVisible` when drawn with this `HLook`.

**See Also:**

`getPreferredSize()`, `setDefaultSize(Dimension)`

**hitTest(HAdjustmentValue, Point)**

```
public int hitTest(HAdjustmentValue component, java.awt.Point pt)
```

Returns a value which indicates the pointer click position in the on-screen representation of the adjustable component. Note that it is a valid implementation option to always return `ADJUST_NONE`.

**Specified By:**

`hitTest(HAdjustmentValue, Point)` in interface `HAdjustableLook`

**Parameters:**

`component` - the `HAdjustmentValue` component for which the hit position should be calculated.

`pt` - the pointer click point.

**Returns:**

one of `ADJUST_NONE`, `ADJUST_BUTTON_LESS`, `ADJUST_PAGE_LESS`, `ADJUST_THUMB`, `ADJUST_PAGE_MORE` or `ADJUST_BUTTON_MORE`.

**isOpaque(HVisible)**

```
public boolean isOpaque(HVisible visible)
```

Returns true if the entire painted area of the `HVisible` when using this look is fully opaque, i.e. the `showLook(Graphics, HVisible, int)` method guarantees that all pixels are painted in an opaque `Color`.

The default value is implementation specific and depends on the background painting mode of the given `HVisible`. The consequences of an invalid overridden value are implementation specific.

**Specified By:**

`isOpaque(HVisible)` in interface `HLook`

**Parameters:**

`visible` - the visible to test

**Returns:**

true if all the pixels with the `java.awt.Component#getBounds` method of an `HVisible` using this look are fully opaque, i.e. the `showLook(Graphics, HVisible, int)` method guarantees that all pixels are painted in an opaque `Color`, otherwise false.

**showLook(Graphics, HVisible, int)**

```
public void showLook(java.awt.Graphics g, HVisible visible, int state)
```

The `showLook(Graphics, HVisible, int)` method is responsible for repainting the entire `HVisible` component, (including any content set on the component, and the component background), subject to the `clipRect` of the `Graphics` object passed to it.

The `showLook(Graphics, HVisible, int)` method should not modify the `clipRect` of the `Graphics` object that is passed to it.

For looks which draw content (e.g. `HTextLook`, `HGraphicLook` and `HAnimateLook`), if no content is associated with the component, the `showLook(Graphics, HVisible, int)` method paints the component with its current background `Color` according to the `setBackgroundMode(int)` method of `HVisible` and draws any (implementation-specific) borders. Note that by default the background mode is set so as to **not** paint a background. Furthermore on platforms which support transparent colors the background `Color` may be partially or completely transparent.

Any resources **explicitly** associated with an `HLook` should be loaded by the `HLook` during its creation, etc., or via its `setXXX()` methods. Note that the "standard" looks don't load content by default.

This method is called from the `paint(Graphics)` method of `HVisible` and must never be called from elsewhere. Components wishing to redraw themselves should call their `repaint` method in the usual way.

**Specified By:**

`showLook(Graphics, HVisible, int)` in interface `HLook`

**Parameters:**

`g` - the graphics context.

`visible` - the visible.

`state` - the state parameter indicates the state of the visible, allowing the look to render the appropriate content for that state. Note that some components (e.g. `HStaticRange`, `HRange`, `HRangeValue`) do not use state-based content).

---

**widgetChanged(HVisible, HChangeData[])**

```
public void widgetChanged(HVisible visible, HChangeData[] changes)
```

Called by the `HVisible` whenever its content, state, or any other data changes. See the class description of `HVisible` for more information about the `changes` parameter.

The implementation of this method should work out which graphical areas of the `HVisible` have changed and make any relevant calls to trigger the repainting of those areas.

A minimum implementation of this method could simply call

```
visible.repaint()
```

**Specified By:**

`widgetChanged(HVisible, HChangeData[])` in interface `HLook`

**Parameters:**

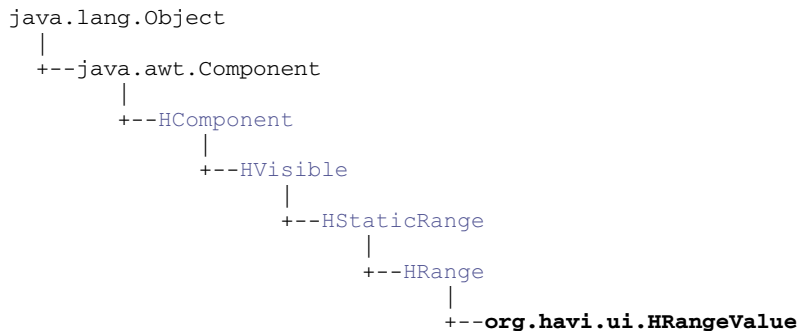
`visible` - the `HVisible` which has changed

`changes` - an array containing hint data and associated hint objects. If this argument is `null` a full repaint will be triggered.

# org.havi.ui HRangeValue

## Syntax

public class HRangeValue extends HRange implements HAdjustmentValue, HAdjustmentInputPreferred



## All Implemented Interfaces:

HAdjustmentInputPreferred, HAdjustmentValue, HMatteLayer, HNavigable, HNavigationInputPreferred, HNoInputPreferred, HOrientable, HState, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, TestOpacity

## Description

The `HRangeValue` is a user interface component used to display a value within a fixed range (as `org.havi.ui.HStaticRange` `HStaticRange`}) which enables a user to navigate to and alter the value of it, i.e. it can have the input focus and it can be adjusted. By default it uses the `HRangeLook` to render itself. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
height	height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
orientation	The "orientation" of the range object.	ORIENT_LEFT_T O_RIGHT	setOrientation( int)	getOrientation( )

minimum	The minimum value that can be returned by this range object.	0	<code>setRange(int, int)</code>	<code>getMinValue()</code>
maximum	The maximum value that can be returned by this range object.	100	<code>setRange(int, int)</code>	<code>getMaxValue()</code>
value	The current value returned by this range object.	0	<code>setValue(int)</code>	<code>getValue()</code>

#### Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte ( <code>HMatte</code> ).	none (i.e. <code>getMatte()</code> returns null)	<code>setMatte(HMatte)</code>	<code>getMatte()</code>
The text layout manager responsible for text formatting.	An <code>HDefaultTextLayoutManager</code> object.	<code>setTextLayoutManager(HTextLayoutManager)</code>	<code>getTextLayoutManager()</code>
The background painting mode	<code>NO_BACKGROUND_FILL</code>	<code>setBackgroundMode(int)</code>	<code>getBackgroundMode()</code>
The default preferred size	not set (i.e. null) unless specified by width and height parameters	<code>setDefaultSize(Dimension)</code>	<code>getDefaultSize()</code>
The horizontal content alignment	<code>HALIGN_CENTER</code>	<code>setHorizontalAlignment(int)</code>	<code>getHorizontalAlignment()</code>
The vertical content alignment	<code>VALIGN_CENTER</code>	<code>setVerticalAlignment(int)</code>	<code>getVerticalAlignment()</code>
The content scaling mode	<code>RESIZE_NONE</code>	<code>setResizeMode(int)</code>	<code>getResizeMode()</code>
The default "look" for this class.	A platform specific <code>HRangeLook</code>	<code>setDefaultLook(HRangeLook)</code>	<code>getDefaultLook()</code>
The "look" for this object.	The <code>HRangeLook</code> returned from <code>HRangeValue.getDefaultLook</code> when this object was created.	<code>setLook(HLook)</code>	<code>getLook()</code>
The offsets for the "thumb" of this range control	min = 0, max = 0	<code>setThumbOffsets(int, int)</code>	<code>getThumbMinOffset()</code> / <code>getThumbMaxOffset()</code>
The behavior of this range object with respect to its "thumb" values	<code>SLIDER_BEHAVIOR</code>	<code>setBehavior(int)</code>	<code>getBehavior()</code>



The block increment for this object.	1 unit	<code>setBlockIncrement(int)</code>	<code>getBlockIncrement()</code>
The gain focus sound.	null	<code>setGainFocusSound(HSound)</code>	<code>getGainFocusSound()</code>
The lose focus sound.	null	<code>setLoseFocusSound(HSound)</code>	<code>getLoseFocusSound()</code>

**See Also:**

`HStaticRange`, `HRange`, `HNavigable`, `HAdjustmentValue`

## Constructors

---

**HRangeValue()**

```
public HRangeValue()
```

Creates an `HRangeValue` object. See the class description for details of constructor parameters and default values.

---

**HRangeValue(int, int, int, int)**

```
public HRangeValue(int orientation, int minimum, int maximum, int value)
```

Creates an `HRangeValue` object. See the class description for details of constructor parameters and default values.

---

**HRangeValue(int, int, int, int, int, int, int, int)**

```
public HRangeValue(int orientation, int minimum, int maximum, int value, int x, int y,
                  int width, int height)
```

Creates an `HRangeValue` object. See the class description for details of constructor parameters and default values.

## Methods

---

**addAdjustmentListener(HAdjustmentListener)**

```
public void addAdjustmentListener(HAdjustmentListener l)
```

Adds the specified `HAdjustmentListener` to receive `HAdjustmentEvent` sent from this object. If the listener has already been added further calls will add further references to the listener, which will then receive multiple copies of a single event.

**Specified By:**

`addAdjustmentListener(HAdjustmentListener)` in interface `HAdjustmentValue`

**Parameters:**

`l` - the `HAdjustmentListener` to be notified.

---

**getAdjustmentSound()**

```
public HSound getAdjustmentSound()
```

Get the sound to be played when the value changes.

**Specified By:**

```
getAdjustmentSound() in interface HAdjustmentValue
```

**Returns:**

The sound played when the value changes

---

**getAdjustMode()**

```
public boolean getAdjustMode()
```

Get the adjustment mode for this `HRangeValue` . If the returned value is `true` the component is in adjustment mode, and its value may be changed on receipt of `ADJUST_LESS` and `ADJUST_MORE` events.

The component is switched into and out of adjustment mode on receiving `ADJUST_START_CHANGE` and `ADJUST_END_CHANGE` events.

**Specified By:**

```
getAdjustMode() in interface HAdjustmentInputPreferred
```

**Returns:**

true if this component is in adjustment mode, false otherwise.

---

**getBlockIncrement()**

```
public int getBlockIncrement()
```

Get the block increment for this `HRangeValue` .

**Specified By:**

```
getBlockIncrement() in interface HAdjustmentValue
```

**Returns:**

the block increment value for this `HRangeValue` .

---

**getDefaultLook()**

```
public static HRangeLook getDefaultLook()
```

Returns the currently set default look for `HRangeValue` components.

**Returns:**

The look that is used by default when creating a new `HRangeValue` component.

---

**getGainFocusSound()**

```
public HSound getGainFocusSound()
```

Get the sound associated with the gain focus event.

**Specified By:**

```
getGainFocusSound() in interface HNavigable
```

**Overrides:**

```
getGainFocusSound() in class HRange
```

**Returns:**

The sound played when the component gains focus. If no sound is associated with gaining focus, then null shall be returned.

---

**getLoseFocusSound()**

```
public HSound getLoseFocusSound()
```

Get the sound associated with the lost focus event.

**Specified By:**

```
getLoseFocusSound() in interface HNavigable
```

**Overrides:**

```
getLoseFocusSound() in class HRange
```

**Returns:**

The sound played when the component loses focus. If no sound is associated with losing focus, then null shall be returned.

---

**getMove(int)**

```
public HNavigable getMove(int keyCode)
```

Provides the `HNavigable` object that is navigated to when a particular key is pressed.

**Specified By:**

```
getMove(int) in interface HNavigable
```

**Overrides:**

```
getMove(int) in class HRange
```

**Parameters:**

`keyCode` - The key code of the pressed key.

**Returns:**

Returns the `HNavigable` object, or if no `HNavigable` is associated with the `keyCode` then returns `null`.

---

**getNavigationKeys()**

```
public int[] getNavigationKeys()
```

Retrieve the set of key codes which this component maps to navigation targets.

**Specified By:**

```
getNavigationKeys() in interface HNavigationInputPreferred
```

**Overrides:**

```
getNavigationKeys() in class HRange
```

**Returns:**

an array of key codes, or `null` if no navigation targets are set on this component.

---

**getUnitIncrement()**

```
public int getUnitIncrement()
```

Get the unit increment for this `HRangeValue`.

**Specified By:**

```
getUnitIncrement() in interface HAdjustmentValue
```

**Returns:**

the increment value for this `HRangeValue`.

---

**isSelected()**

```
public boolean isSelected()
```

Indicates if this component has focus.

**Specified By:**

```
isSelected() in interface HNavigable
```

**Overrides:**

```
isSelected() in class HRange
```

**Returns:**

true if the component has focus, otherwise returns false.

---

**processHAdjustmentEvent(HAdjustmentEvent)**

```
public void processHAdjustmentEvent(HAdjustmentEvent evt)
```

Process an `HAdjustmentEvent` sent to this `HRangeValue`.

**Specified By:**

```
processHAdjustmentEvent(HAdjustmentEvent) in interface  
HAdjustmentInputPreferred
```

**Parameters:**

evt - the `HAdjustmentEvent` to process.

---

**processHFocusEvent(HFocusEvent)**

```
public void processHFocusEvent(HFocusEvent evt)
```

Process an `HFocusEvent` sent to this `HRangeValue`.

**Specified By:**

```
processHFocusEvent(HFocusEvent) in interface HNavigationInputPreferred
```

**Overrides:**

```
processHFocusEvent(HFocusEvent) in class HRange
```

**Parameters:**

evt - the `HFocusEvent` to process.

---

**removeAdjustmentListener(HAdjustmentListener)**

```
public void removeAdjustmentListener(HAdjustmentListener l)
```

Removes the specified `HAdjustmentListener` so that it no longer receives `HAdjustmentEvent` from this object. If the specified listener is not registered, the method has no effect. If multiple references to a single listener have been registered it should be noted that this method will only remove one reference per call.

**Specified By:**

```
removeAdjustmentListener(HAdjustmentListener) in interface HAdjustmentValue
```

**Parameters:**

l - the `HAdjustmentListener` to be removed from notification.

---

**setAdjustmentSound(HSound)**

```
public void setAdjustmentSound(HSound sound)
```

Associate a sound to be played when the value is modified. The sound is played irrespective of whether an `HAdjustmentEvent` is sent to one or more listeners.

**Specified By:**

`setAdjustmentSound(HSound)` in interface `HAdjustmentValue`

**Parameters:**

`sound` - the sound to be played, when the value is modified. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

**setAdjustMode(boolean)**

```
public void setAdjustMode(boolean adjust)
```

Set the adjustment mode for this `HRangeValue`.

This method is provided for the convenience of component implementors. Interoperable applications shall not call this method. It cannot be made protected because interfaces cannot have protected methods.

**Specified By:**

`setAdjustMode(boolean)` in interface `HAdjustmentInputPreferred`

**Parameters:**

`edit` - true to switch this component into adjustment mode, false otherwise.

**See Also:**

`getAdjustMode()`

**setBlockIncrement(int)**

```
public void setBlockIncrement(int increment)
```

Set the block increment for this `HRangeValue`.

**Specified By:**

`setBlockIncrement(int)` in interface `HAdjustmentValue`

**Parameters:**

`increment` - the amount by which the value of the `HRangeValue` should change when an `ADJUST_PAGE_LESS` or `ADJUST_PAGE_MORE` event is received. Values of `increment` less than one shall be treated as a value of one.

**setDefaultLook(HRangeLook)**

```
public static void setDefaultLook(HRangeLook look)
```

Sets the default `HLook` for further `HRangeValue` Components.

**Parameters:**

`look` - The `HLook` that will be used by default when creating a new `HRangeValue` component. Note that this parameter may be null, in which case newly created components shall not draw themselves until a non-null look is set using the `setLook(HLook)` method.

**setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)**

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
    HNavigable right)
```

Set the focus control for an `HNavigable` component. Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is a convenience function for application

programmers where a standard up, down, left and right focus traversal between components is required.

Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

Note that this API does not prevent the creation of "isolated" `HNavigable` components --- authors should endeavor to avoid confusing the user.

**Specified By:**

`setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` in interface `HNavigable`

**Overrides:**

`setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` in class `HRange`

**Parameters:**

`up` - The `HNavigable` component to move to, when the user generates a `VK_UP` KeyEvent. If there is no `HNavigable` component to move "up" to, then null should be specified.

`down` - The `HNavigable` component to move to, when the user generates a `VK_DOWN` KeyEvent. If there is no `HNavigable` component to move "down" to, then null should be specified.

`left` - The `HNavigable` component to move to, when the user generates a `VK_LEFT` KeyEvent. If there is no `HNavigable` component to move "left" to, then null should be specified.

`right` - The `HNavigable` component to move to, when the user generates a `VK_RIGHT` KeyEvent. If there is no `HNavigable` component to move "right" to, then null should be specified.

### **setGainFocusSound(HSound)**

```
public void setGainFocusSound(HSound sound)
```

Associate a sound with gaining focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_GAINED`. This sound will start to be played when an object implementing this interface gains focus. It is not guaranteed to be played to completion. If the object implementing this interface loses focus before the audio completes playing, the audio will be truncated. Applications wishing to ensure the audio is always played to completion must implement special logic to slow down the focus transitions.

By default, an `HNavigable` object does not have any gain focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

**Specified By:**

`setGainFocusSound(HSound)` in interface `HNavigable`

**Overrides:**

`setGainFocusSound(HSound)` in class `HRange`

**Parameters:**

`sound` - the sound to be played, when the component gains focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

### **setLoseFocusSound(HSound)**

```
public void setLoseFocusSound(HSound sound)
```

Associate a sound with losing focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_LOST`. This sound will start to be played when an

object implementing this interface loses focus. It is not guaranteed to be played to completion. It is implementation dependent whether and when this sound will be truncated by any gain focus sound played by the next object to gain focus.

By default, an `HNavigable` object does not have any lose focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

**Specified By:**

`setLoseFocusSound(HSound)` in interface `HNavigable`

**Overrides:**

`setLoseFocusSound(HSound)` in class `HRange`

**Parameters:**

`sound` - the sound to be played, when the component loses focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

### **setMove(int, HNavigable)**

```
public void setMove(int keyCode, HNavigable target)
```

Defines the navigation path from the current `HNavigable` to another `HNavigable` when a particular key is pressed.

Note that `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

**Specified By:**

`setMove(int, HNavigable)` in interface `HNavigable`

**Overrides:**

`setMove(int, HNavigable)` in class `HRange`

**Parameters:**

`keyCode` - The key code of the pressed key. Any numerical keycode is allowed, but the platform may not be able to generate all keycodes. Application authors should only use keys for which `HRcCapabilities.isSupported()` or `HKeyCapabilities.isSupported()` returns true.

`target` - The target `HNavigable` object that should be navigated to. If a target is to be removed from a particular navigation path, then `null` should be specified.

### **setUnitIncrement(int)**

```
public void setUnitIncrement(int increment)
```

Set the unit increment for this `HRangeValue`.

**Specified By:**

`setUnitIncrement(int)` in interface `HAdjustmentValue`

**Parameters:**

`increment` - the amount by which the value of the `HRangeValue` should change when an `ADJUST_LESS` or `ADJUST_MORE` event is received. Values of `increment` less than one shall be treated as a value of one.

# org.havi.ui HScene

## Syntax

```
public class HScene extends java.awt.Container implements HComponentOrdering
```

```
java.lang.Object
|
+--java.awt.Component
|
+--java.awt.Container
|
+--org.havi.ui.HScene
```

## All Implemented Interfaces:

`HComponentOrdering`, `java.awt.image.ImageObserver`, `java.awt.MenuContainer`,  
`java.io.Serializable`

## Description

An `HScene` is a container representing the displayable area on-screen within which the application can display itself and thus interact with the user.

`HScene` may be regarded as a simple connection to the window management policy within the device, acting as a "screen resource reservation mechanism" denoting the area within which an application may present components.

### Rendering Behavior

By default, `HScene` does not paint itself on-screen, only its added "child" components and hence its only immediate graphical effect is to "clip" its child components. However, it is possible to request that the entire `HScene` be painted in the current background color before any drawing takes place, and/or that a background image be drawn before the children are rendered.

### Class Behavior

For all interoperable applications, the `HScene` is considered the top-level component of the application. No parent component to an `HScene` should be accessible to applications. Interoperable applications should not use the `getParent` method in `HScene`, since results are implementation dependent and valid implementations may generate a run-time error.

Although `HScene` is a subclass of `java.awt.Container`, implementations are allowed to insert extra classes in the inheritance tree between `HScene` and `Container`. It is allowed that this may result in `HScene` inheriting additional methods beyond those specified here. This allows platforms with only one native `java.awt.Frame` to use `HScene` as specified, whereas platforms with support for multiple `java.awt.Frame` or `java.awt.Window` classes can use an `HScene` class derived from the appropriate class.

`HScene` follow the design pattern of the `java.awt.Window` class. They are not a scarce resource on the platform. On platforms which only support one `HScene` being visible at one time the current `HScene` both loses the input focus and is hidden (e.g. iconified) when another application successfully requests the input focus. Two `java.awt.event.WindowEvent` events, with ids `WINDOW_DEACTIVATED` and `WINDOW_ICONIFIED`, shall be generated and sent to the `HScene` which has lost the focus and the `isShowing` method for that `HScene` shall return false.

In terms of delegation, the `HScene` shall behave like a `Window` with a native peer implementation, in that it will not appear to delegate any functionality to any parent object. Components which do not specify default characteristics inherit default values transitively from their parent objects. Therefore, the



implementation of `HScene` must have valid defaults defined for all characteristics, e.g. `Font`, `foreground Color`, `background Color`, `ColorModel`, `Cursor` and `Locale`.

Additional Z-order support

`HScene` extends the `java.awt.Container` class by providing additional Z-ordering capabilities, which are required since components in the HAVi user-interface are explicitly allowed to overlap each other. The Z-ordering capabilities are defined by the `HComponentOrdering` interface.

Note that these Z-ordering capabilities (`addBefore`, `addAfter`, `pop`, `popInFrontOf`, `popToFront`, `push`, `pushBehind` and `pushToBack`) must be implemented by (implicitly) reordering the child Components within the `HScene`, so that the standard AWT convention that the Z-order is defined as the order in which Components are added to a given Container is maintained. See the description for `HComponentOrdering` for more details.

Shortcut Keys

It is an implementation option for HAVi systems to implement shortcut keys by forwarding `java.awt.KeyEvent` events to the parent `HScene` of an application. Under these circumstances it is the responsibility of the application designer to ensure that the relevant `java.awt.KeyEvent` events used for "shortcut keys" are not consumed by any custom components.

Implementations of the standard HAVi UI components which process `java.awt.event.KeyEvent` events shall not consume any `KeyEvent`, thus allowing their use as a shortcut key on implementation which rely on `KeyEvents` being available in this way.

Event Handling

The mechanism by which input events are passed to the `HScene` and its component hierarchy is not specified.

Note that whether the `HScene` is visible or not (as determined by the `isVisible()` method) does not guarantee that it has the input focus and is receiving events.

When the application initially gains the input focus, this is indicated by the system sending a `java.awt.event.WindowEvent` of type `WINDOW_ACTIVATED` to the `HScene`. The `HScene` should request that a child component gain the focus, if one is available. However, the mechanism by which this occurs is intentionally not specified here.

When the entire application loses the user's focus, the system shall notify the `HScene` that it is no longer receiving events by sending a `java.awt.event.WindowEvent` of type `WINDOW_DEACTIVATED` to the `HScene`.

If the entire application has its location, or extent modified, then these events should be signaled via a `java.awt.event.WindowEvent` with an id of `ComponentEvent.COMPONENT_MOVED` being sent to the `HScene` object.

Acquiring and Displaying HScenes

There is no public constructor for `HScene`, it is constructed by an `HSceneFactory`. Only one `HScene` per `HGraphicsDevice` can be acquired at any one time for each application.

The application may request that it be made visible by calling the `show()` method. This method should ensure that the `HScene` is completely visible to the user, e.g. by expanding an icon, or changing the stacking order between competing overlapping applications.

Making the `HScene` visible shall not automatically cause it to receive or even request input focus. Input focus can be requested by the application by calling the `requestFocus` method at any time.

The `requestFocus` method should first ensure that the `HScene` is visible, and call the `show()` method if this is not the case. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Visibility of the <code>HScene</code>	false	<code>setVisible</code>	<code>isVisible</code>
Activity of associated shortcuts	Shortcuts are active	<code>enableShortcuts(boolean)</code>	<code>isEnabledShortcuts()</code>
Associated layout manager	null	<code>java.awt.Container#setLayout</code>	<code>java.awt.Container#getLayout</code>
Background image mode	<code>IMAGE_NONE</code>	<code>setRenderMode(int)</code>	<code>getRenderMode()</code>

## Fields

---

### BACKGROUND\_FILL

```
public static final int BACKGROUND_FILL
```

A constant for use with the `setBackgroundMode(int)` method which indicates that the `HScene` should fill its bounding rectangle with its current background color before painting any background image and children.

---

### IMAGE\_CENTER

```
public static final int IMAGE_CENTER
```

A constant for use with `setRenderMode(int)` which specifies that any background image rendered in this `HScene` should be centered in the extents of the `HScene`. No tiling or scaling of the image shall be performed.

---

### IMAGE\_NONE

```
public static final int IMAGE_NONE
```

A constant for use with `setRenderMode(int)` which specifies that a background image should not be rendered in this `HScene`.

---

### IMAGE\_STRETCH

```
public static final int IMAGE_STRETCH
```

A constant for use with `setRenderMode(int)` which specifies that any background image rendered in this `HScene` should be stretched to fill the extents of the `HScene`.

---

## IMAGE\_TILE

```
public static final int IMAGE_TILE
```

A constant for use with `setRenderMode(int)` which specifies that any background image rendered in this `HScene` should be tiled starting from the top left origin of the `HScene`. No scaling of the image shall be performed.

---

## NO\_BACKGROUND\_FILL

```
public static final int NO_BACKGROUND_FILL
```

A constant for use with the `setBackgroundMode(int)` method which indicates that the `HScene` should not fill its bounding rectangle with any color before painting any background image and children.

---

## Constructors

---

### HScene()

```
protected HScene()
```

It is not intended that applications should directly construct `HScene` objects. `HScene` objects should be constructed via the `HSceneFactory` classes factory methods.

Creates an `HScene` object. See the class description for details of constructor parameters and default values.

**See Also:**

`HSceneFactory`

---

## Methods

---

### addAfter(Component, Component)

```
public java.awt.Component addAfter(java.awt.Component component, java.awt.Component front)
```

Adds a `java.awt.Component` to this `HScene` directly behind a previously added `java.awt.Component`.

If `component` has already been added to this container, then `addAfter` moves `component` behind `front`. If `front` and `component` are the same component which was already added to this container, `addAfter` does not change the ordering of the components and returns `component`.

This method affects the Z-order of the `java.awt.Component` children within the `HScene`, and may also implicitly change the numeric ordering of those children.

**Specified By:**

`addAfter(Component, Component)` in interface `HComponentOrdering`

**Parameters:**

`component` - is the `java.awt.Component` to be added to the `HScene`

`front` - is the `java.awt.Component`, which `component` will be placed behind, i.e. `front` will be directly in front of the added `java.awt.Component`

**Returns:**

If the `java.awt.Component` is successfully added, then it will be returned from this call. If the `java.awt.Component` is not successfully added, e.g. `front` is not a `java.awt.Component` currently added to the `HScene`, then `null` will be returned.

This method must be implemented in a thread safe manner.

---

### **addBefore(Component, Component)**

```
public java.awt.Component addBefore(java.awt.Component component,
                                   java.awt.Component behind)
```

Adds a `java.awt.Component` to this `HScene` directly in front of a previously added `java.awt.Component`.

If `component` has already been added to this container, then `addBefore` moves `component` in front of `behind`. If `behind` and `component` are the same component which was already added to this container, `addBefore` does not change the ordering of the components and returns `component`.

This method affects the Z-order of the `java.awt.Component` children within the `HScene`, and may also implicitly change the numeric ordering of those children.

#### **Specified By:**

`addBefore(Component, Component)` in interface `HComponentOrdering`

#### **Parameters:**

`component` - is the `java.awt.Component` to be added to the `HScene`

`behind` - is the `java.awt.Component`, which `component` will be placed in front of, i.e. `behind` will be directly behind the added `java.awt.Component`

#### **Returns:**

If the `java.awt.Component` is successfully added, then it will be returned from this call. If the `java.awt.Component` is not successfully added, e.g. `behind` is not a `java.awt.Component` currently added to the `HScene`, then `null` will be returned.

This method must be implemented in a thread safe manner.

---

### **addShortcut(int, HActionable)**

```
public boolean addShortcut(int keyCode, HActionable comp)
```

Adds a shortcut key to action the specified `HActionable`. Generating the defined `java.awt.KeyEvent` or `HRcEvent` keycode causes the specified component to become actioned --- potentially any `keyCode` may have a shortcut associated with it. The shortcut will only be added if the `HActionable` component is a child component in the container hierarchy of which the `HScene` is the root container. If this is not the case this method shall return false.

Note that a maximum of one `HActionable` may be associated with a given `keyCode`. An `HActionable` can have at most one associated shortcut keycode. Calling `addShortcut(int, HActionable)` repeatedly with the same `HActionable` will result in the previous short cut being removed. A short cut can be set on an invisible `HActionable` and therefore it is possible to provide short-cuts that have no user representation.

If the relevant `keyCode` is received by the `HScene`, then the `HActionable` will be actioned by the `HScene` sending it an `ACTION_PERFORMED`.

#### **Parameters:**

`keyCode` - the keycode that represents the short cut. If keycode is `java.awt.event.KeyEvent#VK_UNDEFINED`, then the shortcut will not be added, any existing shortcut for this component will not be changed and this method shall return false.

`comp` - The actionable component that will be actioned.

**Returns:**

true if the shortcut was added, false otherwise.

---

**addWindowListener(WindowListener)**

```
public void addWindowListener(java.awt.event.WindowListener wl)
```

Add a listener to receive any `java.awt.event.WindowEvents` sent from this `HScene`. If the listener has already been added further calls will add further references to the listener, which will then receive multiple copies of a single event.

**Parameters:**

`wl` - The `java.awt.event.WindowListener` to be notified of any `java.awt.event.WindowEvents`.

---

**dispose()**

```
public void dispose()
```

Disposes of this window. This method must be called at application exit to release the resources that are used for the `HScene`. Calling this method will be equivalent to calling `dispose(HScene)`. Calling this method on an `HScene` that was already disposed will have no effect or result in an unspecified exception being thrown. This is intentionally implementation-dependent.

---

**enableShortcuts(boolean)**

```
public void enableShortcuts(boolean enable)
```

Enables or disables all short cuts that are currently set on the Scene. To enable or disable a single shortcut use `addShortcut(int, HActionable)` or `removeShortcut(int)`.

Note `enableShortcuts(boolean)` does not remove existing added `HScene` shortcuts - they are merely disabled and may be subsequently re-enabled with `enableShortcuts(boolean)`.

**Parameters:**

`enable` - a value of true indicates all shortcuts are to be enabled, and a value of false indicates all shortcuts are to be disabled.

---

**getAllShortcutKeycodes()**

```
public int[] getAllShortcutKeycodes()
```

Returns all keycodes added in the `HScene` as shortcuts.

**Returns:**

all keycodes added in the `HScene` as shortcuts, there are no ordering guarantees.

---

**getBackgroundImage()**

```
public java.awt.Image getBackgroundImage()
```

Retrieve any image used as a background for this `HScene`.

**Returns:**

an image used as a background, or `null` if no image is set. Note that depending on the current render mode any image set may not actually be rendered.

**See Also:**

`setRenderMode(int)`

---

**getBackgroundMode()**

```
public int getBackgroundMode()
```

Get the background mode of this `HScene`. The return value specifies whether the paint method should draw the background (i.e. a rectangle filling the bounds of the `HScene`).

**Returns:**

one of `NO_BACKGROUND_FILL` or `BACKGROUND_FILL`.

---

**getFocusOwner()**

```
public java.awt.Component getFocusOwner()
```

Returns the child component of this `HScene` which has focus if and only if this `HScene` is active.

**Returns:**

the component with focus, or null if no children have focus assigned to them.

---

**getPixelCoordinatesHScreenRectangle(Rectangle)**

```
public HScreenRectangle getPixelCoordinatesHScreenRectangle(java.awt.Rectangle r)
```

Returns an `HScreenRectangle` which corresponds to the graphics (AWT) pixel area specified by the parameter in this `HScene`. (i.e. within the `HScene`'s coordinate space).

**Parameters:**

`r` - the AWT pixel area within this `HScene` (i.e. within the `HScene`'s coordinate space), specified as an `java.awt.Rectangle`.

**Returns:**

an `HScreenRectangle` which corresponds to the graphics (AWT) pixel area specified by the parameter in this `HScene` (i.e. within the `HScene`'s coordinate space).

---

**getRenderMode()**

```
public int getRenderMode()
```

Get the rendering mode of any background image associated with this `HScene`.

**Returns:**

the rendering mode, one of `IMAGE_NONE`, `IMAGE_STRETCH`, `IMAGE_CENTER` or `IMAGE_TILE`.

---

**getSceneTemplate()**

```
public HSceneTemplate getSceneTemplate()
```

Return an `HSceneTemplate` describing this `HScene`. This template can be queried in order to obtain the size & position of the `HScene` in screen coordinates and the display device used for the `HScene`, etc.

**Returns:**

an `HSceneTemplate` describing of the `HScene`.

---

**getShortcutComponent(int)**

```
public HActionable getShortcutComponent(int keyCode)
```

Retrieve the `HActionable` associated with the specified shortcut key.

---

**Parameters:**

`keyCode` - the shortcut key code to be queried for an associated `HActionable` .

**Returns:**

the `HActionable` associated with the specified key if `keyCode` is a valid shortcut key for this `HScene` , null otherwise.

---

**getShortcutKeyCode(HActionable)**

```
public int getShortcutKeyCode(HActionable comp)
```

Returns the keycode associated with the specified `HActionable` component.

**Parameters:**

`comp` - the `HActionable` to return the keycode that it is associated with.

**Returns:**

the keycode associated with the specified `HActionable` component, if it is currently a valid shortcut "target", otherwise return `java.awt.event.KeyEvent#VK_UNDEFINED`.

---

**isDoubleBuffered()**

```
public boolean isDoubleBuffered()
```

Returns `true` if all the drawing done during the update and paint methods for this specific `HScene` object is automatically double buffered.

**Overrides:**

`java.awt.Component.isDoubleBuffered()` in class `java.awt.Component`

**Returns:**

`true` if all the drawing done during the update and paint methods for this specific `HScene` object is automatically double buffered, or `false` if drawing is not double buffered. The default value for the double buffering setting is platform-specific.

---

**isEnabledShortcuts()**

```
public boolean isEnabledShortcuts()
```

Returns the status of all short cuts that are currently set on the `HScene`.

**Returns:**

`true` if shortcuts are enabled, `false` otherwise.

**See Also:**

`enableShortcuts(boolean)`

---

**isOpaque()**

```
public boolean isOpaque()
```

Returns `true` if the entire `HScene` area, as given by the `java.awt.Component#getBounds` method, is fully opaque, i.e. its paint method (or surrogate methods) guarantee that all pixels are painted in an opaque `Color`.

By default, the return value depends on the value of the current background mode, as set by the `setBackgroundMode(int)` method. The return value should be overridden by subclasses that can guarantee full opacity. The consequences of an invalid overridden value are implementation specific.

**Returns:**

`true` if all the pixels within the area given by the `java.awt.Component#getBounds` method are fully opaque, i.e. its paint method (or surrogate methods) guarantee that all pixels are painted in an opaque `Color`, otherwise `false`.

---

### isVisible()

```
public boolean isVisible()
```

Determines if the `HScene` (or more properly its added child components) is Visible. Initially an `HScene` is invisible.

**Overrides:**

`java.awt.Component.isVisible()` in class `java.awt.Component`

**Returns:**

`true` if the `HScene` is visible; `false` otherwise.

---

### paint(Graphics)

```
public void paint(java.awt.Graphics g)
```

`HScene` objects override the paint method (defined in `java.awt.Component`) to paint the added "child" components on top of an optional background color or image. The paint behavior is as follows:

1. If the current background mode is `BACKGROUND_FILL`, the entire `HScene` is first filled using the current background color.
2. If a background image has been set using the `setBackgroundImage(Image)` method, *and* the current image rendering mode as set using `setRenderMode(int)` is not `IMAGE_NONE`, the specified image is painted. Scaling and tiling are performed according to the render mode set.
3. Finally any children of the `HScene` are rendered in z-order.

**Overrides:**

`java.awt.Container.paint(java.awt.Graphics)` in class `java.awt.Container`

**Parameters:**

`g` - the graphics context to use for painting.

---

### pop(Component)

```
public boolean pop(java.awt.Component component)
```

Moves the specified `java.awt.Component` one component nearer in the Z-order, i.e. swapping it with the `java.awt.Component` that was directly in front of it.

If `component` is already at the front of the Z-order, the order is unchanged and `pop` returns `true`.

**Specified By:**

`pop(Component)` in interface `HComponentOrdering`

**Parameters:**

`component` - The `java.awt.Component` to be moved.

**Returns:**

returns `true` on success, `false` on failure, for example if the `java.awt.Component` has yet to be added to the `HScene`.

---

### popInFrontOf(Component, Component)

```
public boolean popInFrontOf(java.awt.Component move, java.awt.Component behind)
```



Puts the specified `java.awt.Component` in front of another `java.awt.Component` in the Z-order of this `HScene` .

If `move` and `behind` are the same component which has been added to the container `popInFront` does not change the Z-order and returns `true`.

**Specified By:**

`popInFrontOf(Component, Component)` in interface `HComponentOrdering`

**Parameters:**

`move` - The `java.awt.Component` to be moved directly in front of the "behind" Component in the Z-order of this `HScene` .

`behind` - The `java.awt.Component` which the "move" Component should be placed directly in front of.

**Returns:**

returns `true` on success, `false` on failure, for example when either `java.awt.Component` has yet to be added to the `HScene` . If this method fails, the Z-order is unchanged.

### **popToFront(Component)**

```
public boolean popToFront(java.awt.Component component)
```

Brings the specified `java.awt.Component` to the "front" of the Z-order in this `HScene` .

If component is already at the front of the Z-order, the order is unchanged and `popToFront` returns `true`.

**Specified By:**

`popToFront(Component)` in interface `HComponentOrdering`

**Parameters:**

`component` - The `java.awt.Component` to bring to the "front" of the Z-order of this `HScene` .

**Returns:**

returns `true` on success, `false` on failure, for example when the `java.awt.Component` has yet to be added to the `HScene` . If this method fails, the Z-order is unchanged.

### **processWindowEvent(WindowEvent)**

```
protected void processWindowEvent(java.awt.event.WindowEvent we)
```

Process a `java.awt.event.WindowEvent` for this `HScene` .

**Parameters:**

`we` - the `java.awt.event.WindowEvent` to be processed.

### **push(Component)**

```
public boolean push(java.awt.Component component)
```

Moves the specified `java.awt.Component` one component further away in the Z-order, i.e. swapping it with the `java.awt.Component` that was directly behind it.

If component is already at the back of the Z-order, the order is unchanged and `push` returns `true`.

**Specified By:**

`push(Component)` in interface `HComponentOrdering`

**Parameters:**

`component` - The `java.awt.Component` to be moved.

**Returns:**

returns `true` on success, `false` on failure, for example if the `java.awt.Component` has yet to be added to the `HScene`.

---

**pushBehind(Component, Component)**

```
public boolean pushBehind(java.awt.Component move, java.awt.Component front)
```

Puts the specified `java.awt.Component` behind another `java.awt.Component` in the Z-order of this `HScene`.

If `move` and `front` are the same component which has been added to the container `pushBehind` does not change the Z-order and returns `true`.

**Specified By:**

`pushBehind(Component, Component)` in interface `HComponentOrdering`

**Parameters:**

`move` - The `java.awt.Component` to be moved directly behind the "front" Component in the Z-order of this `HScene`.

`front` - The `java.awt.Component` which the "move" Component should be placed directly behind.

**Returns:**

returns `true` on success, `false` on failure, for example when either `java.awt.Component` has yet to be added to the `HScene`.

---

**pushToBack(Component)**

```
public boolean pushToBack(java.awt.Component component)
```

Place the specified `java.awt.Component` at the "back" of the Z-order in this `HScene`.

If `component` is already at the back the Z-order is unchanged and `pushToBack` returns `true`.

**Specified By:**

`pushToBack(Component)` in interface `HComponentOrdering`

**Parameters:**

`component` - The `java.awt.Component` to place at the "back" of the Z-order of this `HScene`.

**Returns:**

returns `true` on success, `false` on failure, for example when the `java.awt.Component` has yet to be added to the `HScene`. If the component was not added to the container `pushToBack` does not change the Z-order.

---

**removeShortcut(int)**

```
public void removeShortcut(int keyCode)
```

Removes the specified short-cut key. if the specified short-cut key is not registered, the method has no effect

**Parameters:**

`keyCode` - The keycode that represents the short cut

---

**removeWindowListener(WindowListener)**

```
public void removeWindowListener(java.awt.event.WindowListener wl)
```

Remove a listener so that it no longer receives any `java.awt.event.WindowEvents`. If the specified listener is not registered, the method has no effect. If multiple references to a single listener have been registered it should be noted that this method will only remove one reference per call.

**Parameters:**

`wl` - The `java.awt.event.WindowListener` to be removed from notification of any `java.awt.event.WindowEvents`.

---

**setBackgroundImage(Image)**

```
public void setBackgroundImage(java.awt.Image image)
```

Set an image which shall be painted in the background of the `HScene`, after the background has been drawn according to the current mode set with `setBackgroundMode(int)`, but before any children are drawn. The image is rendered according to the current render mode set with `setRenderMode(int)`.

Note that the use of a background image in this way may affect the return value of the `isOpaque()` method, depending on the image and the current rendering mode.

**Parameters:**

`image` - the image to be used as a background. If this parameter is `null` any current image is removed. Note that depending on the current render mode any image set may not actually be rendered.

**See Also:**

`setRenderMode(int)`

---

**setBackgroundMode(int)**

```
public void setBackgroundMode(int mode)
```

Set the background mode of this `HScene`. The value specifies whether the paint method should draw the background (i.e. a rectangle filling the bounds of the `HScene`).

Note that the background mode will affect the return value of the `isOpaque()` method, depending on the value of the `mode` parameter. A fill mode of `BACKGROUND_FILL` implies that `isOpaque()` must return `true`.

**Parameters:**

`mode` - one of `NO_BACKGROUND_FILL` or `BACKGROUND_FILL`. If `mode` is not a valid value, an `IllegalArgumentException` will be thrown.

---

**setRenderMode(int)**

```
public boolean setRenderMode(int mode)
```

Set the rendering mode of any background image associated with this `HScene`.

Note that the minimum requirement is to support only the `IMAGE_NONE` mode. Support of the other modes is platform and implementation specific.

**Parameters:**

`mode` - the rendering mode, one of `IMAGE_NONE`, `IMAGE_STRETCH`, `IMAGE_CENTER` or `IMAGE_TILE`.

**Returns:**

`true` if the mode was set successfully, `false` if the mode is not supported by the platform.

---

**setVisible(boolean)**

```
public void setVisible(boolean visible)
```

Shows or hides this `HScene` depending on the value of the input parameter `visible`. An `HScene` is initially not visible, a call to `setVisible` should be used to make it visible. Unlike the `show()` method This method does *not* cause the `HScene` to request the input focus.

**Overrides:**

`java.awt.Component.setVisible(boolean)` in class `java.awt.Component`

**Parameters:**

`visible` - If true, makes this `HScene` visible; otherwise, hides this `HScene` .

**See Also:**

`show()`

---

**show()**

```
public void show()
```

Shows this `HScene` , and brings it to the front.

If this `HScene` is not yet visible, `show` makes it visible. If this `HScene` is already visible, then this method brings it to the front.

**Overrides:**

`java.awt.Component.show()` in class `java.awt.Component`

**See Also:**

`setVisible(boolean)`

# org.havi.ui HSceneFactory

## Syntax

```
public class HSceneFactory

java.lang.Object
|
+--org.havi.ui.HSceneFactory
```

## Description

The `HSceneFactory` class provides a generic mechanism for an application to request `HScene` resources from a (conceptual) window management system. The `HSceneFactory` is the single entry to potentially multiple `HGraphicsDevice` centric window management policies.

The `HSceneFactory` class provides an opaque interface between any application (or window) management scheme and the Java application, itself.

Note that only one `HScene` per `HGraphicsDevice` can be acquired at any one time for each application.

`HScene` follow the design pattern of the `java.awt.Window` class. They are not a scarce resource on the platform. On platforms which only support one `HScene` being visible at one time the current `HScene` both loses the input focus and is hidden (e.g. iconified) when another application successfully requests the input focus. Two `java.awt.event.WindowEvent` events, with ids `WINDOW_DEACTIVATED` and `WINDOW_ICONIFIED`, shall be generated and sent to the `HScene` which has lost the focus and the `isShowing` method for that `HScene` shall return false.

The constraints on the sizing and positioning of the `HScene` returned by the methods are dependent on the platform-specific `HScene` support. The three scenarios are defined below:

### Platforms Supporting a Full Multi-Window System

Platforms where windows may obscure each other shall allow applications to create `HScene` objects which are fully within the area of the supporting `HGraphicsDevice` without any restriction on size or location. Whether successful creation of `HScenes` which are wholly or partially outside the area of the `HGraphicsDevice` is supported is implementation dependent.

### Platforms Supporting a Single Window System

Platforms supporting a simple "full-screen" view on a single application at any one time are required to allow applications to create `HScenes` which cover the full area of the supporting `HGraphicsDevice`. `HScenes` matching this description shall be returned as the default `HScene` for the supporting `HGraphicsDevice`. It is implementation dependent whether requests to create `HScene` objects which cover less than the full area of the supporting `HGraphicsDevice` succeed and if they succeed, what the consequences are of their being displayed.

### Platforms Supporting a Restricted Multi-Window System

Platforms supporting a "paned" system where each application occupies an area on-screen that is always visible make fewer guarantees to applications. When an `HScene` is created on such a system, the platform shall return an `HScene` which, if visible at that time, would cover as much as possible of the requested area of the supporting `HGraphicsDevice` considering all other visible `HScenes` at that time. `HScenes` which are not visible at this time shall not be considered when fixing the location & size of the new `HScene`. `HScenes` which are visible at this time shall not be effected by the creation of the new `HScene`.

When `setVisible(boolean)` is called on such an `HScene`, the platform shall attempt to make the `HScene` visible using its currently set position and size. If this would conflict with `HScenes` which are already visible (e.g. because of changes between when the `HScene` was created and when

`setVisible` was called) then the call to `setVisible` shall fail silently. Applications are responsible for testing for failure using the `isVisible()` method. The HScenes of already visible applications shall not be impacted by this method call.

The above text specifies the relationship between an `HScene` and its supporting `HGraphicsDevice`. This specification intentionally does not define minimum requirements for `HGraphicsDevices` or for their relationship with other `HScreenDevices` of any type.

Calling `resizeScene(HScene, HSceneTemplate)` for an `HScene` shall apply the same policies as described above for newly created HScenes when deciding whether the resizing operation requested is possible. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Methods

### `dispose(HScene)`

```
public void dispose(HScene scene)
```

This method allows an application to dispose of its `HScene`, indicating that the application has no further need for user interaction (i.e. its resources may be released to the system, for future garbage collection. After `dispose()` has been called the application may then acquire another `HScene`.

After calling this method, further method calls on the `HScene` will have no effect or result in an unspecified exception being thrown. This is intentionally implementation-dependent.

**Parameters:**

`scene` - the `HScene` to be disposed of.

**See Also:**

`HScene`

### `getBestScene(HSceneTemplate)`

```
public HScene getBestScene(HSceneTemplate hst)
```

Returns an `HScene` that best corresponds to the input `HSceneTemplate`, or null if such an `HScene` cannot be generated.

Note that conflict may occur between properties in the `HSceneTemplate` and the `HGraphicsConfigTemplate` corresponding to the currently active `HGraphicsConfiguration`. In the event of conflict between properties which are PREFERRED or UNNECESSARY, the properties concerned shall be ignored and the default will prevail. In the event of conflict between properties which are REQUIRED, this method shall fail and return null.

**Parameters:**

`hst` - the `HSceneTemplate` to match against

**Returns:**

the `HScene` that matches the properties as specified in the `HSceneTemplate`, or null if they cannot be satisfied, or if no further `HScene` are available for this application.

**getBestSceneTemplate(HSceneTemplate)**

```
public HSceneTemplate getBestSceneTemplate(HSceneTemplate hst)
```

Returns an `HSceneTemplate` that is closest to to the input `HSceneTemplate` and corresponds to an `HScene` which could be successfully created on this platform at the time that this method is called.

Note that since some platforms may support more than one concurrent application there is no guarantee that the values returned by this method would actually match those of a subsequently requested `HScene`, using the same template.

Note that conflict may occur between properties in the `HSceneTemplate` and the `HGraphicsConfigTemplate` corresponding to the currently active `HGraphicsConfiguration`. In the event of conflict between properties which are PREFERRED or UNNECESSARY, the properties concerned shall be ignored and the default will prevail. In the event of conflict between properties which are REQUIRED, this method shall fail and return null.

**Parameters:**

`hst` - The `HSceneTemplate` properties that the `HScene` should satisfy.

**Returns:**

an `HSceneTemplate` that best corresponds to the input `HSceneTemplate`.

**getDefaultHScene()**

```
public HScene getDefaultHScene()
```

Create the default `HScene` for the default `HScreen` for this application. This shall be identical to calling

```
org.havi.ui.HScene.getDefaultHScene(org.havi.ui.HScreen.getDefaultHScreen())
```

**Returns:**

the default `HScene` for the default `HScreen`. If the application has already obtained an `HScene` for the default `HScreen`, then that `HScene` is returned.

**getDefaultHScene(HScreen)**

```
public HScene getDefaultHScene(HScreen screen)
```

Create the default `HScene` for this `HScreen`. This shall use the `HGraphicsConfiguration` returned by calling

```
screen.getDefaultHGraphicsDevice().getDefaultConfiguration()
```

**Parameters:**

`screen` - the screen for which the `HScene` should be returned.

**Returns:**

the default `HScene` for this `HScreen`. If the application has already obtained an `HScene` for this `HScreen`, then that `HScene` is returned.

---

**getFullScreenScene(HGraphicsDevice)**

```
public HScene getFullScreenScene(HGraphicsDevice device)
```

Create a full-screen `HScene` on the specified `HGraphicsDevice` or null if such an `HScene` cannot be generated.

**Parameters:**

`device` - the graphics device with which to create the `HScene`. This is obtained through a `HGraphicsConfigTemplate` / `HGraphicsConfiguration` sequence as described in the document for these classes.

**Returns:**

a created full-screen `HScene` or null if this is not possible.

---

**getInstance()**

```
public static HSceneFactory getInstance()
```

Returns an `HSceneFactory` object to an application.

**Returns:**

an `HSceneFactory` object to an application. Note that repeated invocations of this method should return the same object (reference).

---

**resizeScene(HScene, HSceneTemplate)**

```
public HSceneTemplate resizeScene(HScene hs, HSceneTemplate hst)
```

Resizes an `HScene` so that it best corresponds to the input `HSceneTemplate`, or remains unchanged if it cannot be so resized.

**Parameters:**

`hs` - the `HScene` to be resized.

`hst` - the `HSceneTemplate` which denotes the new size / location. Only size location options in the `HSceneTemplate` will be considered.

**Returns:**

an `HSceneTemplate` that indicates the `HScene` properties after (possible) resizing.

**Throws:**

`java.lang.IllegalStateException` - if the `HScene` had previously been disposed.



# org.havi.ui HSceneTemplate

## Syntax

```
public class HSceneTemplate
    java.lang.Object
    |
    +--org.havi.ui.HSceneTemplate
```

## Description

The `HSceneTemplate` class is used to obtain an `HScene` subject to a variety of constraints.

The following constraints are supported:

Preference	Object
<code>GRAPHICS_CONFIGURATION</code>	<code>HGraphicsConfiguration</code>
<code>SCENE_PIXEL_LOCATION</code>	<code>java.awt.Point</code>
<code>SCENE_PIXEL_DIMENSION</code>	<code>java.awt.Dimension</code>
<code>SCENE_SCREEN_LOCATION</code>	<code>HScreenPoint</code>
<code>SCENE_SCREEN_DIMENSION</code>	<code>HScreenDimension</code>

Note that as defined here users must set both a location and a dimension preference to request a given rectangle area. Instances of `HSceneTemplate` returned by the system will always be fully specified. Therefore, if an application only sets a preference on `SCENE_SCREEN_LOCATION` the system will fill in `SCENE_SCREEN_DIMENSION`, `SCENE_PIXEL_LOCATION` and `SCENE_PIXEL_DIMENSION` with default values based on the appropriate `HGraphicsConfiguration` and its associated `HGraphicsDevice`.

In the event of a conflict between `REQUIRED` preferences specified in pixel coordinates (`SCENE_PIXEL_LOCATION`, `SCENE_PIXEL_DIMENSION`) and normalized screen coordinates (`SCENE_SCREEN_LOCATION`, `SCENE_SCREEN_DIMENSION`) the system shall ignore the pixel coordinates and use only the screen coordinate preferences. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

None.

### See Also:

`HSceneFactory`

## Fields

---

### GRAPHICS\_CONFIGURATION

```
public static final int GRAPHICS_CONFIGURATION
```

A value for use in the preference field of the `setPreference(int, Object, int)`, `getPreferenceObject(int)` and `getPreferencePriority(int)` methods in the `HSceneTemplate` that indicates that the `HScene` be created with a specified `HGraphicsConfiguration` (corresponding to a particular `HGraphicsDevice`).

By default the `HSceneTemplate` creates `HScene` on the default `HScreen` default `HGraphicsDevice` with its current `HGraphicsConfiguration`.

---

### LARGEST\_PIXEL\_DIMENSION

```
public static final java.awt.Dimension LARGEST_PIXEL_DIMENSION
```

A `Dimension` object for use in the object field of the `setPreference(int, Object, int)` and `getPreferenceObject(int)` methods in the `HSceneTemplate` that indicates that the `SCENE_PIXEL_DIMENSION` feature should be set to its largest possible dimension.

---

### PREFERRED

```
public static final int PREFERRED
```

A value for use in the priority field of the `setPreference(int, Object, int)` and `getPreferencePriority(int)` methods in the `HSceneTemplate` that indicates that this feature is preferred over a selection that does not include this feature, although both selections can be considered valid.

---

### REQUIRED

```
public static final int REQUIRED
```

A value for use in the priority field of the `setPreference(int, Object, int)` and `getPreferencePriority(int)` methods in the `HSceneTemplate` that indicates that this feature is required in the `HScene`. If this feature is not available, do not create an `HScene` object.

---

### SCENE\_PIXEL\_DIMENSION

```
public static final int SCENE_PIXEL_DIMENSION
```

A value for use in the preference field of the `setPreference(int, Object, int)`, `getPreferenceObject(int)` and `getPreferencePriority(int)` methods in the `HSceneTemplate` that indicates that the `HScene` be created with preferred dimensions in pixels as given by a `Dimension` object. If the `Dimension` object is `LARGEST_DIMENSION` then the returned `HScene` should have the largest possible dimensions.

---

## SCENE\_PIXEL\_LOCATION

```
public static final int SCENE_PIXEL_LOCATION
```

A value for use in the preference field of the `setPreference(int, Object, int)`, `getPreferenceObject(int)` and `getPreferencePriority(int)` methods in the `HSceneTemplate` that indicates that the `HScene` be created with a preferred location in pixels as given by a `Point` object. The graphics pixels shall correspond to the pixel setting for the `HGraphicsDevice` settings as indicated by the `HGraphicsConfiguration` as specified in the `HSceneTemplate` (or its default value).

---

## SCENE\_SCREEN\_DIMENSION

```
public static final int SCENE_SCREEN_DIMENSION
```

A value for use in the preference field of the `setPreference(int, Object, int)`, `getPreferenceObject(int)` and `getPreferencePriority(int)` methods in the `HSceneTemplate` that indicates that the `HScene` be created with preferred dimensions in normalized screen coordinates as given by an `HScreenDimension` object.

---

## SCENE\_SCREEN\_LOCATION

```
public static final int SCENE_SCREEN_LOCATION
```

A value for use in the preference field of the `setPreference(int, Object, int)`, `getPreferenceObject(int)` and `getPreferencePriority(int)` methods in the `HSceneTemplate` that indicates that the `HScene` be created with a preferred location in normalized screen coordinates, as given by an `HScreenPoint` object.

---

## UNNECESSARY

```
public static final int UNNECESSARY
```

A value for use in the priority field of the `setPreference(int, Object, int)` and `getPreferencePriority(int)` methods in the `HSceneTemplate` that indicates that this feature is unnecessary in the `HScene`. A selection without this feature is preferred over a selection that includes this feature since it is not used.

## Constructors

---

### **HSceneTemplate()**

```
public HSceneTemplate()
```

Creates a new `HSceneTemplate` object. See the class description for details of constructor parameters and default values.

## Methods

---

### **getPreferenceObject(int)**

```
public java.lang.Object getPreferenceObject(int preference)
```

Return the preference object for the specified preference.

**Parameters:**

`preference` - the preference to be indicated.

**Returns:**

the preference object for the specified preference. Valid values shall be of the following types, or null if no preference object was set:

- An `HGraphicsConfiguration` object which is returned for the `GRAPHICS_CONFIGURATION` preference.
- A `java.awt.Point` object which is returned for the `SCENE_PIXEL_LOCATION` preference.
- A `java.awt.Dimension` object which is returned for the `SCENE_PIXEL_DIMENSION` preference.
- An `HScreenPoint` object which is returned for the `SCENE_SCREEN_LOCATION` preference.
- An `HScreenDimension` object which is returned for the `SCENE_SCREEN_DIMENSION` preference.

### **getPreferencePriority(int)**

```
public int getPreferencePriority(int preference)
```

Return the priority for the specified preference.

**Parameters:**

`preference` - the preference to be indicated. Valid values are: `GRAPHICS_CONFIGURATION` , `SCENE_PIXEL_LOCATION` , `SCENE_PIXEL_DIMENSION` , `SCENE_SCREEN_LOCATION` and `SCENE_SCREEN_DIMENSION` .

**Returns:**

the priority for the specified preference.

### **setPreference(int, Object, int)**

```
public void setPreference(int preference, java.lang.Object object, int priority)
```

Set the indicated preference (and associated value object) to have the specified priority. If the preference has been previously set, then the previous object and priority shall be overwritten.

By default, the preferences should have an `UNNECESSARY` priority.

**Parameters:**

`preference` - the preference to be indicated. Valid values are: `GRAPHICS_CONFIGURATION` , `SCENE_PIXEL_LOCATION` , `SCENE_PIXEL_DIMENSION` , `SCENE_SCREEN_LOCATION` and `SCENE_SCREEN_DIMENSION` .

An `IllegalArgumentException` shall be thrown if the preference is not a valid value as listed above.

`object` - the Object associated with the given preference.

An `IllegalArgumentException` shall be thrown if the object is not valid for the preference as specified in the class description.

`priority` - the priority of the preference. Valid values are: `REQUIRED` , `PREFERRED` and `UNNECESSARY` .

If `priority` is not a valid priority as defined here a `java.lang.IllegalArgumentException` will be thrown.

# org.havi.ui HScreen

## Syntax

```
public class HScreen
```

```
java.lang.Object
```

```
|
```

```
+--org.havi.ui.HScreen
```

## Description

This class describes the final output composition of a device. It ties together all the (MPEG) video decoders, all the graphics sub-systems and backgrounds which are all combined together before finally being displayed. A platform with two independent displays would support two instances of this class. Where a device outputs audio closely bound with video, that audio output can also be represented through this class.

Since an HScreen represents a single video output signal from a device, all the devices which contribute to that signal must have certain properties in common. It is not possible to select conflicting configurations for different devices on the same HScreen - for example having a video device whose logical output has a 4:3 picture aspect ratio and a graphics device whose logical output has a 16:9 picture aspect ratio. This specification intentionally does not define configurations, or which configurations would be conflicting, since these are essentially region or market dependent. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Methods

### getBestConfiguration(HBackgroundConfigTemplate[])

```
public HBackgroundConfiguration getBestConfiguration(HBackgroundConfigTemplate[] hbcta)
```

Returns an `HBackgroundConfiguration` from an `HBackgroundDevice` which is present on this `HScreen` that best matches at least one of the specified `HBackgroundConfigTemplate`, or null if this is not possible.

Equally best in this sense means that the configurations satisfy an equal number of preferences with priorities `PREFERRED` and `PREFERRED_NOT` and all preferences with priorities `REQUIRED` and `REQUIRED_NOT`.

Configurations are chosen according to the following algorithm, based on the priority as supplied to `setPreference(int, int)`. Configurations must:

1. satisfy ALL the preferences whose priority was `REQUIRED`
2. satisfy NONE of the preferences whose priority was `REQUIRED_NOT`
3. satisfy as many as possible of the preferences whose priority was `PREFERRED`.
4. satisfy as few as possible of the preferences whose priority was `PREFERRED_NOT`.

Preferences whose priority was `DONT_CARE` are ignored.

This method returns null if no configuration exists that satisfies all `REQUIRED` and `REQUIRED_NOT` priorities.

**Parameters:**

`hvca` - the array of `HBackgroundConfiguration` objects to choose from, represented as an array of `HBackgroundConfiguration` objects.

**Returns:**

an `HBackgroundConfiguration` object that is the best matching configuration possible, or null if no `HBackgroundConfiguration` passes the criteria.

### **getBestConfiguration(HGraphicsConfigTemplate[])**

```
public HGraphicsConfiguration getBestConfiguration(HGraphicsConfigTemplate[] hgcta)
```

Returns an `HGraphicsConfiguration` from an `HGraphicsDevice` which is present on this `HScreen` that best matches at least one of the specified `HGraphicsConfigTemplate`. If this is not possible this method will attempt to construct an `HEmulatedGraphicsConfiguration` where the emulated configuration best matches one of the specified `HGraphicsConfigTemplate`. If this is not possible, null is returned.

Equally best in this sense means that the configurations satisfy an equal number of preferences with priorities `PREFERRED` and `PREFERRED_NOT` and all preferences with priorities `REQUIRED` and `REQUIRED_NOT`.

Configurations are chosen according to the following algorithm, based on the priority as supplied to `setPreference(int, int)`. Configurations must:

1. satisfy ALL the preferences whose priority was `REQUIRED`
2. satisfy NONE of the preferences whose priority was `REQUIRED_NOT`
3. satisfy as many as possible of the preferences whose priority was `PREFERRED`.
4. satisfy as few as possible of the preferences whose priority was `PREFERRED_NOT`.

Preferences whose priority was `DONT_CARE` are ignored.

This method returns null if no configuration exists that satisfies all `REQUIRED` and `REQUIRED_NOT` priorities.

**Parameters:**

`hgcta` - the array of `HGraphicsConfigTemplate` objects to choose from.

**Returns:**

an `HGraphicsConfiguration` object that is the best matching configuration possible, or null if no `HGraphicsConfiguration` or `HEmulatedGraphicsConfiguration` object passes the criteria.

### **getBestConfiguration(HVideoConfigTemplate[])**

```
public HVideoConfiguration getBestConfiguration(HVideoConfigTemplate[] hvcta)
```

Returns an `HVideoConfiguration` from an `HVideoDevice` which is present on this `HScreen` that best matches at least one of the specified `HVideoConfigTemplate`. If this is not possible, null is returned.

Equally best in this sense means that the configurations satisfy an equal number of preferences with priorities `PREFERRED` and `PREFERRED_NOT` and all preferences with priorities `REQUIRED` and `REQUIRED_NOT`.

Configurations are chosen according to the following algorithm, based on the priority as supplied to `setPreference(int, int)`. Configurations must:

1. satisfy ALL the preferences whose priority was `REQUIRED`
2. satisfy NONE of the preferences whose priority was `REQUIRED_NOT`
3. satisfy as many as possible of the preferences whose priority was `PREFERRED`.
4. satisfy as few as possible of the preferences whose priority was `PREFERRED_NOT`.

Preferences whose priority was `DONT_CARE` are ignored.

This method returns null if no configuration exists that satisfies all `REQUIRED` and `REQUIRED_NOT` priorities.

**Parameters:**

`hvcta` - - the array of `HVideoConfigTemplate` objects to choose from.

**Returns:**

an `HVideoConfiguration` object that is the best matching configuration possible, or null if no `HVideoConfiguration` object passes the criteria.

### **getCoherentScreenConfigurations(HScreenConfigTemplate[])**

```
public HScreenConfiguration[] getCoherentScreenConfigurations(HScreenConfigTemplate[]
    hscta)
```

Return a coherent set of `HScreenConfiguration` matching a set of templates. One `HScreenConfiguration` will be returned for each `HScreenConfigTemplate` provided as input. The class of the returned objects will correspond to the class of the templates provided as input - where an `HGraphicsConfigTemplate` is provided as input, an `HGraphicsConfiguration` shall be returned. Where an `HVideoConfigTemplate` is provided as input, an `HVideoConfiguration` shall be returned. If more than one template of the same type is provided then the configurations returned must be on different devices but presented on the same screen. If more templates of one type are provided than there are devices of that type in the system, this function will return null.

Coherent means that all the required properties are respected in all of the templates provided and that a configuration can be returned for each template provided.

Conflicts between templates are resolved as discussed in the description of `HScreenConfigTemplate`.

**Parameters:**

`hscta` - an array of objects describing desired / required configurations. If a zero-length array is passed this function will throw a `java.lang.IllegalArgumentException`.

**Returns:**

an array of non-null objects describing a coherent set of screen device configurations or null if no such coherent set is possible.

### **getDefaultHBackgroundDevice()**

```
public HBackgroundDevice getDefaultHBackgroundDevice()
```

Return the default background device for this screen.

**Returns:**

the default background device for this screen or null if none exist.

---

**getDefaultHGraphicsDevice()**

```
public HGraphicsDevice getDefaultHGraphicsDevice()
```

Return the default graphics device for this screen. Note that the `HGraphicsDevice` is the default device for rendering graphics, but it may not be capable of displaying video / mixing it with graphics concurrently.

**Returns:**

the default graphics device for this screen or null if none exist.

---

**getDefaultHScreen()**

```
public static HScreen getDefaultHScreen()
```

Returns the default `HScreen` for this application. For systems where an application is associated with audio or video which is started before the application starts, this method will return the `HScreen` where that associated audio / video is being output.

**Returns:**

the default `HScreen` for this application.

---

**getDefaultHVideoDevice()**

```
public HVideoDevice getDefaultHVideoDevice()
```

Return the default video device for this screen. Note that the `HVideoDevice` is the default device for rendering video, but it may not be capable of displaying graphics / mixing it with graphics concurrently.

**Returns:**

an `HVideoDevice` object or null if none exist.

---

**getHBackgroundDevices()**

```
public HBackgroundDevice[] getHBackgroundDevices()
```

Returns a list of background devices for this screen.

**Returns:**

an array of `HBackgroundDevice` or null if none exist.

---

**getHGraphicsDevices()**

```
public HGraphicsDevice[] getHGraphicsDevices()
```

Returns a list of graphics devices for this screen.

**Returns:**

an array of `HGraphicsDevice` or null if none exist.

---

**getHScreens()**

```
public static HScreen[] getHScreens()
```

Returns all `HScreen` in this system.

---



**Returns:**

an array of `HScreen` representing all `HScreen` in this system.

---

**getHVideoDevices()**

```
public HVideoDevice[] getHVideoDevices()
```

Returns a list of video device for this screen. For systems where an application is associated with video started before the application starts, the first entry in the array returned will be the video device where that video is being output.

**Returns:**

an array of `HVideoDevice` objects or null if none exist.

---

**setCoherentScreenConfigurations(HScreenConfiguration[])**

```
public boolean setCoherentScreenConfigurations(HScreenConfiguration[] hsca)
```

Modify the settings for a set of `HScreenDevice`, based on their `HScreenConfiguration` supplied. Settings should be modified atomically (where possible) or should not be modified if the `HScreenConfiguration` can be determined to be conflicting a priori, i.e. are not "coherent", or would cause an exception to be thrown.

**Parameters:**

`hsca` - the array of configurations that should be applied atomically (where possible). If the length of this array is zero a `java.lang.IllegalArgumentException` will be thrown.

**Returns:**

A boolean indicating whether all `HScreenConfiguration` could be applied successfully. If all of the `HScreenConfiguration` could not be applied successfully, the configuration after this method may not match the configuration of the devices prior to this method being called --- applications should take steps to determine whether a partial change of settings has been made on each device.

**Throws:**

`java.lang.SecurityException` - if the application does not have sufficient rights to set the `HScreenConfiguration` for any of the devices.

`HPermissionDeniedException` - if the application does not currently have the right to set the configuration for any of the devices.

`HConfigurationException` - if the specified `HScreenConfiguration` array is not valid for any of the devices.

# org.havi.ui

# HScreenConfigTemplate

## Syntax

```
public abstract class HScreenConfigTemplate
    java.lang.Object
    |
    +--org.havi.ui.HScreenConfigTemplate
```

## Direct Known Subclasses:

[HBackgroundConfigTemplate](#), [HGraphicsConfigTemplate](#), [HVideoConfigTemplate](#)

## Description

This class describes a configuration of a screen device in terms of various properties and their importance to the application. It is used to request a valid instance of a configuration conforming to the description provided. Sub-classes of this define additional constants which may be used for additional properties. Those classes can be sub-classed in turn to add further properties by systems using the HAVi UI.

In the event of conflicting preferences between HScreenConfigTemplates (for example where a configuration is requested which conflicts with an existing configuration) the following rules shall be applied in all cases.

- Where all the conflicting preferences have priority `PREFERRED`, `PREFERRED_NOT` or `DONT_CARE`, the conflicting preferences shall be discarded from the template.
- Where all the conflicting preferences have priority `REQUIRED` or `REQUIRED_NOT`, the method concerned shall fail.
- In situations other than the above, all preferences with priority `PREFERRED`, `PREFERRED_NOT` or `DONT_CARE` shall be discarded. If there is still a conflict after this, the method concerned shall fail.

The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## See Also:

[HBackgroundConfigTemplate](#), [HGraphicsConfigTemplate](#), [HVideoConfigTemplate](#)

## Fields

---

## DONT\_CARE

```
public static final int DONT_CARE
```

A value for use in the priority field of the `setPreference(int, int)` and `getPreferencePriority(int)` methods in the `HScreenConfigTemplate` that indicates that the presence or absence of this feature in the `HScreenConfiguration` does not matter. Preferences with this priority play no part in the selection procedure.

---

## FLICKER\_FILTERING

```
public static final int FLICKER_FILTERING
```

A value for use in the preference field of the `setPreference(int, int)` and `getPreferencePriority(int)` methods in the `HScreenConfigTemplate` that indicates that the device configuration supports flicker filtering (if it supports an interlaced screen).

---

## INTERLACED\_DISPLAY

```
public static final int INTERLACED_DISPLAY
```

A value for use in the preference field of the `setPreference(int, int)` and `getPreferencePriority(int)` methods in the `HScreenConfigTemplate` that indicates that the device configuration supports an interlaced display.

---

## PIXEL\_ASPECT\_RATIO

```
public static final int PIXEL_ASPECT_RATIO
```

A value for use in the preference field of the `setPreference(int, int)`, `getPreferenceObject(int)` and `getPreferencePriority(int)` methods in the `HScreenConfigTemplate` that indicates that the device configuration supports the pixel aspect ratio, as specified in a `Dimension` object which indicates the (relative) x, y pixel aspect ratio.

Instances of `HScreenConfigTemplate` generated by the platform and returned to applications (e.g. from `getConfigTemplate()`) shall have this preference set to a platform specific value with the `REQUIRED` priority.

---

## PIXEL\_RESOLUTION

```
public static final int PIXEL_RESOLUTION
```

A value for use in the preference field of the `setPreference(int, int)`, `getPreferenceObject(int)` and `getPreferencePriority(int)` methods in the `HScreenConfigTemplate` that indicates that the device configuration supports the pixel resolution, as specified in a `Dimension` object which indicates the pixel resolution of (the area of) the graphics device (as specified using the `SCREEN_RECTANGLE` preference).

Instances of `HScreenConfigTemplate` generated by the platform and returned to applications (e.g. from `getConfigTemplate()`) shall have this preference set to a platform specific value with the `REQUIRED` priority.

---

## PREFERRED

```
public static final int PREFERRED
```

A value for use in the priority field of the `setPreference(int, int)` and `getPreferencePriority(int)` methods in the `HScreenConfigTemplate` that indicates that

this feature is desired in the `HScreenConfiguration`. A selection with this feature is preferred over a selection that does not include this feature, although both selections can be considered valid matches.

---

## PREFERRED\_NOT

```
public static final int PREFERRED_NOT
```

A value for use in the priority field of the `setPreference(int, int)` and `getPreferencePriority(int)` methods in the `HScreenConfigTemplate` that indicates that this feature is desired not to be present in the `HScreenConfiguration`. A selection without this feature is preferred over a selection that does not include this feature, although both selections can be considered valid matches.

---

## REQUIRED

```
public static final int REQUIRED
```

A value for use in the priority field of the `setPreference(int, int)` and `getPreferencePriority(int)` methods in the `HScreenConfigTemplate` that indicates that this feature is required in the `HScreenConfiguration`. If this feature is not available, do not select the `HScreenConfiguration` object.

---

## REQUIRED\_NOT

```
public static final int REQUIRED_NOT
```

A value for use in the priority field of the `setPreference(int, int)` and `getPreferencePriority(int)` methods in the `HScreenConfigTemplate` that indicates that this feature is required not to be present in the `HScreenConfiguration`. If this feature is available, do not select the `HScreenConfiguration` object.

---

## SCREEN\_RECTANGLE

```
public static final int SCREEN_RECTANGLE
```

A value for use in the preference field of the `setPreference(int, int)`, `getPreferenceObject(int)` and `getPreferencePriority(int)` methods in the `HScreenConfigTemplate` that indicates that the device configuration supports presentation on a particular on-screen area, as specified in an `HScreenRectangle` object

Use of this value with an `HScreenRectangle` object other than the identity rectangle `[ 0, 0 -> 1, 1 ]` implies scaling of the presentation to fit within the specified `HScreenRectangle`.

Platforms not supporting such scaling at all or choosing to expose such scaling through other APIs should only return `HScreenConfigTemplate` with the `HScreenRectangle [ 0, 0 -> 1, 1 ]` for the `SCREEN_RECTANGLE` preference.

Instances of `HScreenConfigTemplate` generated by the platform and returned to applications (e.g. from `getConfigTemplate()`) shall have this preference set to a platform specific value with the `REQUIRED` priority.

---

## VIDEO\_GRAPHICS\_PIXEL\_ALIGNED

```
public static final int VIDEO_GRAPHICS_PIXEL_ALIGNED
```

A value for use in the preference field of the `setPreference(int, int)`, `getPreferenceObject(int)` and `getPreferencePriority(int)` methods in the

`HScreenConfigTemplate` that indicates that the device configuration supports the display of video streams and graphics with aligned pixels of the same size. Alignment of the origins of the two pixel coordinate spaces is explicitly not required. Where a video device is moving the video relative to the screen in real time (e.g. implementing pan and scan), graphics configurations shall only support this feature where the implementation of the graphics device can track the position changes in the video device automatically.

If this preference is set and used to request an `HVideoConfiguration` then an `HGraphicsConfiguration` shall be used as the preference object. If it is set and used to request an `HGraphicsConfiguration` then an `HVideoConfiguration` shall be used as the preference object. Requesting an `HVideoConfiguration` which is `VIDEO_GRAPHICS_PIXEL_ALIGNED` with another `HVideoConfiguration` shall fail unless the system concerned supports two `HVideoDevice` objects which can support this exact feature.

This preference is used by the platform as a constraint in selecting configurations. Templates generated by the platform and then returned to applications (e.g. from a `getConfigTemplate` method) shall not have this preference filled in by the platform.

---

## ZERO\_BACKGROUND\_IMPACT

```
public static final int ZERO_BACKGROUND_IMPACT
```

A value for use in the preference field of the `setPreference(int, int)` and `getPreferencePriority(int)` methods in the `HScreenConfigTemplate` that indicates that the device configuration should have zero impact on already displayed backgrounds. If used with the `REQUIRED` priority, this means no changes shall be made. If used with the `PREFERRED` priority, this means changes may be made but should be minimized.

The `PREFERRED_NOT` and `REQUIRED_NOT` priorities may be ignored in the selection of an `HScreenConfiguration` for this preference type.

This preference is used by the platform as a constraint in selecting configurations. Templates generated by the platform and then returned to applications (e.g. from a `getConfigTemplate` method) shall not have this preference filled in by the platform.

---

## ZERO\_GRAPHICS\_IMPACT

```
public static final int ZERO_GRAPHICS_IMPACT
```

A value for use in the preference field of the `setPreference(int, int)` and `getPreferencePriority(int)` methods in the `HScreenConfigTemplate` that indicates that the device configuration should have zero impact on already running graphical applications. If used with the `REQUIRED` priority, this means no changes shall be made. If used with the `PREFERRED` priority, this means changes may be made but should be minimized.

The `PREFERRED_NOT` and `REQUIRED_NOT` priorities may be ignored in the selection of an `HScreenConfiguration` for this preference type.

This preference is used by the platform as a constraint in selecting configurations. Templates generated by the platform and then returned to applications (e.g. from a `getConfigTemplate` method) shall not have this preference filled in by the platform.

---

## ZERO\_VIDEO\_IMPACT

```
public static final int ZERO_VIDEO_IMPACT
```

A value for use in the preference field of the `setPreference(int, int)` and `getPreferencePriority(int)` methods in the `HScreenConfigTemplate` that indicates that the device configuration should have zero impact on already running video streams. If used with the

`REQUIRED` priority, this means no changes shall be made. If used with the `PREFERRED` priority, this means changes may be made but should be minimized.

The `PREFERRED_NOT` and `REQUIRED_NOT` priorities may be ignored in the selection of an `HScreenConfiguration` for this preference type.

This preference is used by the platform as a constraint in selecting configurations. Templates generated by the platform and then returned to applications (e.g. from a `getConfigTemplate` method) shall not have this preference filled in by the platform.

## Constructors

---

### `HScreenConfigTemplate()`

```
public HScreenConfigTemplate()
```

Creates an `HScreenConfigTemplate` object. See the class description for details of constructor parameters and default values.

## Methods

---

### `getPreferenceObject(int)`

```
public java.lang.Object getPreferenceObject(int preference)
```

Return the preference object for the specified preference.

Instances of `HScreenConfigTemplate` which have not had this preference set shall return null for this object. Note that instances constructed by the platform and returned to applications are required to have all preferences (except where explicitly identified) set by the platform before it is returned.

**Parameters:**

`preference` - the preference to be indicated. Valid values for an `HScreenConfigTemplate` are: `VIDEO_GRAPHICS_PIXEL_ALIGNED`, `PIXEL_ASPECT_RATIO`, `PIXEL_RESOLUTION` and `SCREEN_RECTANGLE`.

Subclasses may add further valid values. A `IllegalArgumentException` shall be thrown if the preference is not a valid value for this instance of `HScreenConfigTemplate`, or if `preference` does not have an associated value object.

**Returns:**

the preference object for the specified preference.

---

### `getPreferencePriority(int)`

```
public int getPreferencePriority(int preference)
```

Return the priority for the specified preference.

By default the preferences in a template returned from the system will have an `DONT_CARE` priority unless specified otherwise. Any configuration always satisfies these attributes.

**Parameters:**

`preference` - the preference to be indicated. Valid values for an `HScreenConfigTemplate` are: `ZERO_BACKGROUND_IMPACT`, `ZERO_GRAPHICS_IMPACT`, `ZERO_VIDEO_IMPACT`,

INTERLACED\_DISPLAY , FLICKER\_FILTERING , VIDEO\_GRAPHICS\_PIXEL\_ALIGNED ,  
PIXEL\_ASPECT\_RATIO , PIXEL\_RESOLUTION and SCREEN\_RECTANGLE .

Subclasses may add further valid values. An `IllegalArgumentException` shall be thrown if the preference is not a valid value for this instance of `HScreenConfigTemplate`

**Returns:**

the priority for the specified preference.

**setPreference(int, int)**

```
public void setPreference(int preference, int priority)
```

Set the indicated preference to have the specified priority. If the preference has been previously set, then the previous priority for the preference shall be overwritten.

Attributes that are not filled in in a template (through `setPreference(int, int)` ), shall have the priority `DONT_CARE` . Any configuration always satisfies these attributes.

**Parameters:**

`preference` - the preference to be indicated. Valid values for an `HScreenConfigTemplate` are: `ZERO_GRAPHICS_IMPACT` , `ZERO_BACKGROUND_IMPACT` , `ZERO_VIDEO_IMPACT` , `INTERLACED_DISPLAY` and `FLICKER_FILTERING` .

Subclasses may add further valid values. An `IllegalArgumentException` shall be thrown if the preference is not a valid value for this instance of `HScreenConfigTemplate`

`priority` - the priority of the preference. Valid values are: `REQUIRED` , `PREFERRED` , `DONT_CARE` , `PREFERRED_NOT` and `REQUIRED_NOT` .

If `priority` is not a valid priority as defined here a `java.lang.IllegalArgumentException` will be thrown.

**setPreference(int, Object, int)**

```
public void setPreference(int preference, java.lang.Object object, int priority)
```

Set the indicated preference (and associated value object) to have the specified priority. If the preference has been previously set, then the previous object and priority shall be overwritten.

Attributes that are not filled in in a template (through `setPreference(int, int)` ), shall have the priority `DONT_CARE` . Any configuration always satisfies these attributes.

An application which wishes to remove a preference from an existing template (e.g. one generated by the platform) may call this method with null for the object parameter.

**Parameters:**

`preference` - the preference to be indicated. Valid values for an `HScreenConfigTemplate` are: `VIDEO_GRAPHICS_PIXEL_ALIGNED` , `PIXEL_ASPECT_RATIO` , `PIXEL_RESOLUTION` and `SCREEN_RECTANGLE` .

Subclasses may add further valid values. An `IllegalArgumentException` shall be thrown if the preference is not a valid value for this instance of `HScreenConfigTemplate`

`object` - the Object associated with the given preference, or null.

`priority` - the priority of the preference. Valid values include: `REQUIRED` , `PREFERRED` , `DONT_CARE` , `PREFERRED_NOT` and `REQUIRED_NOT` .

If `priority` is not a valid priority as defined here a `java.lang.IllegalArgumentException` will be thrown.

# org.havi.ui HScreenConfiguration

## Syntax

```
public abstract class HScreenConfiguration  
  
java.lang.Object  
|  
+--org.havi.ui.HScreenConfiguration
```

## Direct Known Subclasses:

[HBackgroundConfiguration](#), [HGraphicsConfiguration](#), [HVideoConfiguration](#)

## Description

The [HScreenConfiguration](#) class describes the characteristics (settings) of an [HScreenDevice](#). There can be many [HScreenConfiguration](#) objects associated with a single [HScreenDevice](#).

## See Also:

[HScreenDevice](#)

## Methods

---

### convertTo(HScreenConfiguration, Point)

```
public java.awt.Point convertTo(HScreenConfiguration destination, java.awt.Point source)
```

Convert a pixel position from one coordinate system to another without including any rounding errors from passing through normalized coordinates. This returns null if this transformation isn't possible for various reasons. These reasons are:

- at least one of the two [HScreenConfiguration](#) isn't pixel based or doesn't yet have a fixed location on the HScreen.
- a non-linear transformation is in use between the two.
- the information needed to calculate this isn't available.
- the transformation is changing with time (e.g. due to pan & scan).

The source position is interpreted in the coordinate system of the [HScreenConfiguration](#) object on which this method is called.

#### Parameters:

`destination` - the destination [HScreenConfiguration](#).

`source` - the pixel position in this [HScreenConfiguration](#).

#### Returns:

the position of the specified pixel position measured in the destination coordinate system, or null if this isn't possible.

---

### getFlickerFilter()

```
public boolean getFlickerFilter()
```

Return whether this configuration includes filtering to reduce interlace flicker.

#### Returns:



true if filtering is included, false otherwise.

---

**getInterlaced()**

```
public boolean getInterlaced()
```

Return whether this configuration is interlaced

**Returns:**

true if this configuration is interlaced, false otherwise.

---

**getOffset(HScreenConfiguration)**

```
public java.awt.Dimension getOffset(HScreenConfiguration hsc)
```

Returns the offset between the origin of the pixel coordinate space of the specified `HScreenConfiguration`, and the origin of the current pixel coordinate space of this `HScreenConfiguration`. The offset is returned in the pixel coordinate space of this `HScreenConfiguration`.

**Parameters:**

`hsc` - the `HScreenConfiguration` to which the offset between pixel origins should be recovered.

**Returns:**

the offset between the pixel coordinate space of the specified `HScreenConfiguration` and the current pixel coordinate space of this `HScreenConfiguration`. A null object will be returned if there is insufficient information to recover the pixel offset.

---

**getPixelAspectRatio()**

```
public java.awt.Dimension getPixelAspectRatio()
```

Return the pixel aspect ratio of this configuration. Some examples are {16:15}, {64:45}, {1:1}.

**Returns:**

the aspect ratio of the pixels in this configuration.

---

**getPixelResolution()**

```
public java.awt.Dimension getPixelResolution()
```

Return the resolution of this configuration in pixels. The pixel coordinate system used is that of the device concerned.

**Returns:**

the resolution of this configuration in pixels.

---

**getScreenArea()**

```
public HScreenRectangle getScreenArea()
```

Return the position and size of this configuration on the screen in screen coordinates.

**Returns:**

the area on the screen of this configuration in screen coordinates.

# org.havi.ui HScreenDevice

## Syntax

```
public class HScreenDevice implements org.davic.resources.ResourceProxy,
    org.davic.resources.ResourceServer
```

```
java.lang.Object
|
+--org.havi.ui.HScreenDevice
```

## Direct Known Subclasses:

[HBackgroundDevice](#), [HGraphicsDevice](#), [HVideoDevice](#)

## All Implemented Interfaces:

[org.davic.resources.ResourceProxy](#), [org.davic.resources.ResourceServer](#)

## Description

An instance of the [HScreen](#) class represents a single independent video output signal from a device. Devices with multiple independent video output signals should support multiple instances of this class. A video output signal is created by adding together the contributions from the devices represented by a number of objects inheriting from the [HScreenDevice](#) class. These can be [HGraphicsDevice](#) objects, [HVideoDevice](#) objects and [HBackgroundDevice](#) objects. A given [HScreen](#) may support any number of any of these objects as far as the API is concerned however some form of profiling may restrict this. In reality right now, one instance of each is all that may reasonably be expected to be present.

Each [HScreenDevice](#) can have multiple settings ([HScreenConfiguration](#)) but only one "setting" ([HScreenConfiguration](#)) can be active at any point in time. The current configuration can be determined on the [HScreenDevice](#) subclasses using their specific `getCurrentConfiguration` methods. The current configuration can be modified on the [HScreenDevice](#) subclasses using their specific `setCurrentConfiguration` methods (assuming sufficient rights, etc.).

Applications may select the best of these configurations for them by creating an instance of [HScreenConfigTemplate](#) and populating that with a number preferences each with a priority. The implementation then matches this template against the range of possible configurations and attempts to find one which matches the template provided. Priorities [REQUIRED](#) and [REQUIRED\\_NOT](#) must be respected. If they cannot be respected then the method call shall fail and not return any configuration. Priorities [PREFERRED](#) and [PREFERRED\\_NOT](#) should be respected as much as possible. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
-----------	-------------	---------------	------------	------------

None.

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

None.

## Methods

---

### **addResourceStatusEventListener(ResourceStatusListener)**

```
public void addResourceStatusEventListener(org.davic.resources.ResourceStatusListener listener)
```

Register a listener for events about changes in the state of the ownership of this device. If the listener has already been added further calls will add further references to the listener, which will then receive multiple copies of a single event.

**Specified By:**

org.davic.resources.ResourceServer.addResourceStatusEventListener(org.davic.resources.ResourceStatusListener) in interface org.davic.resources.ResourceServer

**Parameters:**

listener - the object to be informed of state changes

**See Also:**

HScreenDeviceReleasedEvent, HScreenDeviceReservedEvent

---

### **addScreenConfigurationListener(HScreenConfigurationListener)**

```
public void addScreenConfigurationListener(HScreenConfigurationListener hscl)
```

Add an `HScreenConfigurationListener` to this device, which is notified whenever the device's configuration is modified. If the listener has already been added further calls will add further references to the listener, which will then receive multiple copies of a single event.

**Parameters:**

hscl - the `HScreenConfigurationListener` to be added to this device.

---

### **addScreenConfigurationListener(HScreenConfigurationListener, HScreenConfigTemplate)**

```
public void addScreenConfigurationListener(HScreenConfigurationListener hscl, HScreenConfigTemplate hsct)
```

Add an `HScreenConfigurationListener` to this device, which is notified when the device's configuration is further modified so that it is no longer compatible with the specified `HScreenConfigTemplate`. If the listener has already been added further calls will add further references to the listener, which will then receive multiple copies of a single event.

Note that if the device configuration does not match the specified template, then the listener should be added and a `HScreenConfigurationEvent` immediately generated for the specified `HScreenConfigurationListener`.

**Parameters:**

hscl - the `HScreenConfigurationListener` to be added to this device.

hsct - the `HScreenConfigTemplate` which is to be used to determine compatibility with the device configuration.

---

### **getClient()**

```
public org.davic.resources.ResourceClient getClient()
```

Return the last `ResourceClient` passed to the last successful call to the `reserveDevice(ResourceClient)` method of this instance of `HScreenDevice`, or null if this method has not been called on this instance.

**Specified By:**

org.davic.resources.ResourceProxy.getClient() in interface org.davic.resources.ResourceProxy

**Returns:**

a representation of the intended owner of the resource or null if none has been set.

---

**getIDstring()**

```
public java.lang.String getIDstring()
```

Returns the identification string associated with this [HScreenDevice](#) .

**Returns:**

an identification string

---

**getScreenAspectRatio()**

```
public java.awt.Dimension getScreenAspectRatio()
```

Return the aspect ratio of the screen as far as is known. i.e. 4:3, 16:9, etc.

This [Dimension](#) may be used to determine the pixel aspect ratio for given [HScreenConfiguration](#) .

**Returns:**

a [Dimension](#) object specifying the aspect ratio of the screen

---

**releaseDevice()**

```
public void releaseDevice()
```

Release the right to control of this device. If this application doesn't have this right then this method has no effect. It is not specified whether any device configuration set by this application will be removed from display immediately or whether it will remain on display until a subsequent application obtains the device and sets its own configuration. Applications wishing to ensure a configuration they have installed is removed must actively remove it before calling this method.

---

**removeResourceStatusEventListener(ResourceStatusListener)**

```
public void removeResourceStatusEventListener(org.davic.resources.ResourceStatusListener listener)
```

Remove a listener for events about changes in the state of the ownership of this device. This method has no effect if the listener specified is not registered.

**Specified By:**

org.davic.resources.ResourceServer.removeResourceStatusEventListener(org.davic.resources.ResourceStatusListener) in interface org.davic.resources.ResourceServer

**Parameters:**

listener - the object which is no longer interested

**See Also:**

[HScreenDeviceReleasedEvent](#), [HScreenDeviceReservedEvent](#)

---

**removeScreenConfigurationListener(HScreenConfigurationListener)**

```
public void removeScreenConfigurationListener(HScreenConfigurationListener hsc1)
```

Remove an [HScreenConfigurationListener](#) from this device. if the specified listener is not registered, the method has no effect. If multiple references to a single listener have been registered it should be noted that this method will only remove one reference per call.

**Parameters:**

`hscl` - the `HScreenConfigurationListener` to be removed from this device.

---

**reserveDevice(ResourceClient)**

```
public boolean reserveDevice(org.davic.resources.ResourceClient client)
```

Requests the right to call any method which may otherwise throw an `HPermissionDeniedException`. If this method returns true this exception will never be thrown until this right is revoked as notified by methods on `ResourceClient`. The policy by which the platform decides whether or not to grant this right is not defined in this specification.

Note that the word "right" in this context has nothing to do with security. See the description of `HPermissionDeniedException`.

Once the right to control this device has been granted and not removed in the intervening period further calls to this method shall have no effect and return true.

**Parameters:**

`client` - a representation of the intended owner of the resource

**Returns:**

true if the right is granted, otherwise false

# org.havi.ui

# HScreenDimension

## Syntax

```
public class HScreenDimension
{
    java.lang.Object
    |
    +--org.havi.ui.HScreenDimension
}
```

## Description

`HScreenDimension` denotes a screen dimension expressed as a relative value of the screen dimensions. Note that since these are relative dimensions they are effectively independent of any particular screen's physical dimensions, or aspect ratio.

Note that the extents of the dimension must be positive (including zero), but are otherwise unconstrained - and hence may denote areas greater in size than the entire screen.

Hence,

- (1.0, 1.0) denotes the size of the entire screen.
- (0.5, 0.5) denotes a quarter of the screen.

Note that in practice, particularly in the case of television, the precise dimension may vary slightly due to effects of overscan, etc.

Note that systems using `HScreenDimension` directly should consider the effects of rounding errors, etc. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
width	The horizontal extent of the dimension	no default constructor exists	<code>set-Size(float, float)</code>	---
height	The vertical extent of the dimension	no default constructor exists	<code>set-Size(float, float)</code>	---

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## See Also:

`HScreenPoint`

## Fields

---

**height**

```
public float height
```

---

**width**

```
public float width
```

## Constructors

---

**HScreenDimension(float, float)**

```
public HScreenDimension(float width, float height)
```

Creates an `HScreenDimension` object. See the class description for details of constructor parameters and default values.

## Methods

---

**setSize(float, float)**

```
public void setSize(float width, float height)
```

Set the extents of the `HScreenDimension`.

**Parameters:**

`width` - the horizontal extent of the `HScreenDimension`

`height` - the vertical extent of the `HScreenDimension`

# org.havi.ui HScreenPoint

## Syntax

```
public class HScreenPoint
{
    java.lang.Object
    |
    +--org.havi.ui.HScreenPoint
}
```

## Description

`HScreenPoint` denotes a screen location expressed as a relative value of the screen dimensions. Note that since these are relative dimensions they are effectively independent of any particular screen's physical dimensions, or aspect ratio.

The x coordinate is in terms of the ratio of the particular horizontal screen location to the entire screen width.

The y coordinate is in terms of the ratio of the particular vertical screen location to the entire screen width.

All measurements should be taken from the top, left corner of the screen, measuring positive dimensions down and to the right.

Note that x and y coordinates are not constrained - they may be negative, or have values greater than one - and hence, may denote locations that are not "on- screen".

Hence,

- (0.0, 0.0) denotes the top, left hand corner of the screen.
- (1.0, 0.0) denotes the top, right hand corner of the screen.
- (0.5, 0.5) denotes the center (middle) of the screen.
- (0.0, 1.0) denotes the bottom, left hand corner of the screen.
- (1.0, 1.0) denotes the bottom, right hand corner of the screen.

Note that in practice, particularly in the case of television, the precise location may vary slightly due to effects of overscan, etc. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	The horizontal position of the point	no default constructor exists	<code>setLocation(float, float)</code>	---
y	The vertical position of the point	no default constructor exists	<code>setLocation(float, float)</code>	---

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			



**See Also:**

[HScreenRectangle](#)

## Fields

---

**x**

```
public float x
```

---

**y**

```
public float y
```

## Constructors

---

**HScreenPoint(float, float)**

```
public HScreenPoint(float x, float y)
```

Creates an [HScreenPoint](#) object. See the class description for details of constructor parameters and default values.

## Methods

---

**setLocation(float, float)**

```
public void setLocation(float x, float y)
```

Set the location of the [HScreenPoint](#).

**Parameters:**

*x* - the horizontal position of the point

*y* - the vertical position of the point

# org.havi.ui HScreenRectangle

## Syntax

```
public class HScreenRectangle
{
    java.lang.Object
    |
    +--org.havi.ui.HScreenRectangle
}
```

## Description

`HScreenRectangle` denotes a screen area expressed as a relative value of the screen dimensions. Note that since these are relative dimensions they are effectively independent of any particular screen's physical dimensions, or aspect ratio.

Note that the x and y offset coordinates of the top, left corner of the area are not constrained - they may be negative, or have values greater than one - and hence, may denote an offset location that is not "on-screen". The width and height of the area should be positive (including zero), but are otherwise unconstrained - and hence may denote areas greater in size than the entire screen.

Hence,

- (0.0, 0.0, 1.0, 1.0) denotes the whole of the screen.
- (0.0, 0.0, 0.5, 0.5) denotes the top, left hand quarter of the screen.
- (0.5, 0.0, 0.5, 0.5) denotes the top, right hand quarter of the screen.
- (0.25, 0.25, 0.5, 0.5) denotes a centered quarter-screen area of the screen.
- (0.0, 0.5, 0.5, 0.5) denotes the bottom, left hand quarter of the screen.
- (0.5, 0.5, 0.5, 0.5) denotes the bottom, right hand quarter of the screen.

Note that in practice, particularly in the case of television, the precise location may vary slightly due to effects of overscan, etc.

Note that systems using `HScreenRectangle` directly should consider the effects of rounding errors, etc. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	The horizontal position of the top left corner	no default constructor exists	<code>setLocation(float, float)</code>	---
y	The vertical position of the top left corner	no default constructor exists	<code>setLocation(float, float)</code>	---
width	The width of the rectangle	no default constructor exists	<code>setSize(float, float)</code>	---
height	The height of the rectangle	no default constructor exists	<code>setSize(float, float)</code>	---

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

---

None.

### See Also:

[HScreenPoint](#)

## Fields

---

### height

```
public float height
```

---

### width

```
public float width
```

---

### x

```
public float x
```

---

### y

```
public float y
```

---

## Constructors

---

### HScreenRectangle(float, float, float, float)

```
public HScreenRectangle(float x, float y, float width, float height)
```

Creates an [HScreenRectangle](#) object. See the class description for details of constructor parameters and default values.

## Methods

---

### setLocation(float, float)

```
public void setLocation(float x, float y)
```

Set the location of the top left corner of the [HScreenRectangle](#).

#### Parameters:

x - the horizontal position of the top left corner

y - the vertical position of the top left corner

---

**setSize(float, float)**

```
public void setSize(float width, float height)
```

Set the size of the HScreenRectangle.

**Parameters:**

width - the width of the HScreenRectangle

height - the height of the HScreenRectangle

# org.havi.ui HSelectionInputPreferred

## Syntax

```
public interface HSelectionInputPreferred extends HOrientable
```

## All Known Subinterfaces:

```
HItemValue
```

## All Superinterfaces:

```
HOrientable
```

## Description

A component which implements `HSelectionInputPreferred` indicates that this component expects to receive `HItemEvent` input events.

All interoperable implementations of the `HSelectionInputPreferred` interface must extend `HComponent`.

Note that the `java.awt.Component` method `isFocusTraversable` should always return true for a `java.awt.Component` implementing this interface.

## Methods

---

### getSelectionMode()

```
public boolean getSelectionMode()
```

Get the selection mode for this `HSelectionInputPreferred`. If the returned value is true the component is in selection mode, and the selection may be changed.

The component is switched into and out of selection mode on receiving `ITEM_START_CHANGE` and `ITEM_END_CHANGE` events.

#### Returns:

true if this component is in selection mode, false otherwise.

---

### processHItemEvent(HItemEvent)

```
public void processHItemEvent(HItemEvent evt)
```

Process an `HItemEvent` sent to this `HSelectionInputPreferred`.

#### Parameters:

`evt` - the `HItemEvent` to process.

---

### setSelectionMode(boolean)

```
public void setSelectionMode(boolean adjust)
```

Set the selection mode for this `HSelectionInputPreferred`.

This method is provided for the convenience of component implementors. Interoperable applications shall not call this method. It cannot be made protected because interfaces cannot have protected methods.

**Parameters:**

`edit` - true to switch this component into selection mode, false otherwise.

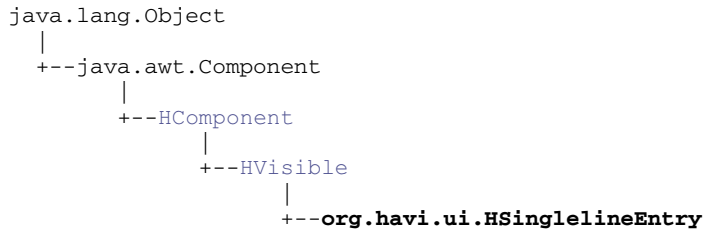
**See Also:**

`getSelectionMode()`

# org.havi.ui HSinglelineEntry

## Syntax

public class HSinglelineEntry extends HVisible implements HTextValue



## Direct Known Subclasses:

HMultilineEntry

## All Implemented Interfaces:

HKeyboardInputPreferred, HMatteLayer, HNavigable, HNavigationInputPreferred, HState, HTextValue, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, TestOpacity

## Description

The `HSinglelineEntry` is a user interface component used to receive a single line of alphanumeric entry from the user and can also be used for password input.

Upon creation the `HSinglelineEntry` is set to a non-editable mode identical in functionality to an `HText`.

On keyboard-based systems, if the user navigates to the component using the keyboard then the component must first be switched into edit mode before it will accept any key presses (other than for navigation to another component). The mechanism by which the component is switched into and out of edit mode is via the `TEXT_START_CHANGE` and `TEXT_END_CHANGE` events, which may be triggered in response to a key stroke or other Java AWT event.

On entering its editable mode the component will send an `TEXT_START_CHANGE` event to all registered `HTextListener` listeners. The `HSinglelineEntry` will then respond to key events by inserting characters into the text string or positioning the insertion point (caret) (via further `HTextEvent` events).

For example, on platforms which do not provide a means of positioning the caret independently from navigating to components, the navigation keys will be interpreted as caret positioning keys in this mode.

While in the editing mode, the component will generate an `TEXT_CHANGE` event whenever the text content of the `HSinglelineEntry` changes (e.g. a character is inserted).

On receiving an `TEXT_END_CHANGE` event the component shall leave its editable mode and send an `TEXT_END_CHANGE` event to all registered `HTextListener` listeners. The user can then navigate out of the `HSinglelineEntry`.

On mouse-based systems, if the user selects the component by clicking a mouse button inside its bounds then the `HSinglelineEntry` will automatically switch into edit mode and generate a `TEXT_START_CHANGE` event. It will stay in edit mode so long as the mouse pointer remains within the bounds of the component. Once the mouse pointer leaves the bounds then it will switch back into non-editable mode and generate a `TEXT_END_CHANGE` event.

By default `HSinglelineEntry` uses the `HSinglelineEntryLook` to render itself. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
width	width of this component in pixels (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
height	height of this component in pixels (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
text	The text within this <code>HSinglelineEntry</code> , to be used as the displayed and editable content for all states.	null	<code>setTextContent(String, int)</code>	<code>getTextContent(int)</code>
max-Chars	The maximum number of characters (per line) allowed in this <code>HSinglelineEntry</code> .	16 characters	<code>setMaxCharsPerLine(int)</code>	<code>getMaxCharsPerLine()</code>
font	The font to be used for this component.	---	<code>java.awt.Component#setFont</code> .	<code>java.awt.Component#getFont</code> .
color	The color to be used for this component.	---	<code>java.awt.Component#setForeground</code> .	<code>java.awt.Component#getForeground</code> .

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte ( <code>HMatte</code> ).	none (i.e. <code>getMatte()</code> returns null)	<code>setMatte(HMatte)</code>	<code>getMatte()</code>
The text layout manager responsible for text formatting.	An <code>HDefaultTextLayoutManager</code> object.	<code>setTextLayoutManager(HTextLayoutManager)</code>	<code>getTextLayoutManager()</code>
The background painting mode	<code>NO_BACKGROUND_FILL</code>	<code>setBackgroundMode(int)</code>	<code>getBackgroundMode()</code>



The default preferred size	not set (i.e. null) unless specified by width and height parameters	<code>setDefaultSize(Dimension)</code>	<code>getDefaultSize()</code>
The horizontal content alignment	<code>HALIGN_CENTER</code>	<code>setHorizontalAlignment(int)</code>	<code>getHorizontalAlignment()</code>
The vertical content alignment	<code>VALIGN_CENTER</code>	<code>setVerticalAlignment(int)</code>	<code>getVerticalAlignment()</code>
The content scaling mode	<code>RESIZE_NONE</code>	<code>setResizeMode(int)</code>	<code>getResizeMode()</code>
Caret position	At the end of the current text string	<code>setCaretCharPosition(int)</code>	<code>getCaretCharPosition()</code>
Input type	<code>INPUT_ANY</code>	<code>setType(int)</code>	<code>getType()</code>
Customized input range	As for <code>INPUT_ALPHANUMERIC</code>	<code>setValidInput(char[])</code>	<code>getValidInput()</code>
Password protection (the echo character)	Entry is "clear", i.e. not password protected.	<code>setEchoChar(char)</code>	<code>getEchoChar()</code> and <code>echoCharIsSet()</code>
The default "look" for this class.	A platform specific <code>HSinglelineEntryLook</code>	<code>setDefaultLook(HSinglelineEntryLook)</code>	<code>getDefaultLook()</code>
The "look" for this object.	The <code>HSinglelineEntryLook</code> returned from <code>HSinglelineEntry.getDefaultLook</code> when this object was created.	<code>setLook(HLook)</code>	<code>getLook()</code>
The gain focus sound.	null	<code>setGainFocusSound(HSound)</code>	<code>getGainFocusSound()</code>
The lose focus sound.	null	<code>setLoseFocusSound(HSound)</code>	<code>getLoseFocusSound()</code>

## Constructors

---

### **HSinglelineEntry()**

```
public HSinglelineEntry()
```

Creates an `HSinglelineEntry` object. See the class description for details of constructor parameters and default values.

---

**HSinglelineEntry(int)**

```
public HSinglelineEntry(int maxChars)
```

Creates an `HSinglelineEntry` object. See the class description for details of constructor parameters and default values.

---

**HSinglelineEntry(int, int, int, int, int)**

```
public HSinglelineEntry(int x, int y, int width, int height, int maxChars)
```

Creates an `HSinglelineEntry` object. See the class description for details of constructor parameters and default values.

---

**HSinglelineEntry(String, int, Font, Color)**

```
public HSinglelineEntry(java.lang.String text, int maxChars, java.awt.Font font,
                        java.awt.Color color)
```

Creates an `HSinglelineEntry` object. See the class description for details of constructor parameters and default values.

---

**HSinglelineEntry(String, int, int, int, int, int, Font, Color)**

```
public HSinglelineEntry(java.lang.String text, int x, int y, int width, int height,
                        int maxChars, java.awt.Font font, java.awt.Color color)
```

Creates an `HSinglelineEntry` object. See the class description for details of constructor parameters and default values.

## Methods

---

**caretNextCharacter()**

```
public void caretNextCharacter()
```

Move the caret to the next character. The caret position is constrained such that the insert point lies within the range as defined by the `getCaretCharPosition()` method.

---

**caretPreviousCharacter()**

```
public void caretPreviousCharacter()
```

Move the caret to the previous character. The caret position is constrained such that the insert point lies within the range as defined by the `getCaretCharPosition()` method.

---

**deleteNextChar()**

```
public boolean deleteNextChar()
```

Delete a character forward of the current caret position.

**Returns:**

true if a character was deleted, false otherwise.

---

**deletePreviousChar()**

```
public boolean deletePreviousChar()
```

Delete a character behind the current caret position.

**Returns:**

true if a character was deleted, false otherwise.

---

**echoCharIsSet()**

```
public boolean echoCharIsSet()
```

Determine if this component has an echo character set, i.e. if the echo character is non-zero.

**Returns:**

true if an echo character is set, false otherwise.

---

**getCaretCharPosition()**

```
public int getCaretCharPosition()
```

Gets the position of the text insertion caret for this the current line in this text component. The valid values of the caret position are from 0 to the length of the string retrieved using `getTextContent(int)`, where 0 implies insertion as the first character (i.e. at the start of the string) and `getTextContent(int)` implies that further characters are to be appended onto the end of the string. Hence, the valid caret positions for the string "abc" of length 3, are 0, 1, 2 and 3 --- with caret locations as shown below:

```
0 "a" 1 "b" 2 "c" 3
```

**Returns:**

the position of the text insertion caret.

---

**getDefaultLook()**

```
public static HSinglelineEntryLook getDefaultLook()
```

Returns the currently set default `HLook` for `HSinglelineEntry` components.

**Returns:**

The `HLook` that is used by default when creating a new `HSinglelineEntry` component.

---

**getEchoChar()**

```
public char getEchoChar()
```

Returns the character to be used for echoing.

**Returns:**

the character to be used for echoing or 0 (ASCII NUL) if no echo character is set.

---

**getEditMode()**

```
public boolean getEditMode()
```

Get the editing mode for this `HSinglelineEntry`. If the returned value is `true` the component is in edit mode, and its textual content may be changed through user interaction such as keyboard events.

---

The component is switched into and out of edit mode on receiving `TEXT_START_CHANGE` and `TEXT_END_CHANGE` events.

**Specified By:**

`getEditMode()` in interface `HKeyboardInputPreferred`

**Returns:**

`true` if this component is in edit mode, `false` otherwise.

---

**getGainFocusSound()**

```
public HSound getGainFocusSound()
```

Get the sound associated with the gain focus event.

**Specified By:**

`getGainFocusSound()` in interface `HNavigable`

**Returns:**

The sound played when the component gains focus. If no sound is associated with gaining focus, then null shall be returned.

---

**getLoseFocusSound()**

```
public HSound getLoseFocusSound()
```

Get the sound associated with the lost focus event.

**Specified By:**

`getLoseFocusSound()` in interface `HNavigable`

**Returns:**

The sound played when the component loses focus. If no sound is associated with losing focus, then null shall be returned.

---

**getMaxCharsPerLine()**

```
public int getMaxCharsPerLine()
```

Get maximum number of characters per (single) line. The behavior of the component when the last character on a line is typed is implementation dependent.

**Returns:**

the maximum number of characters per line.

---

**getMove(int)**

```
public HNavigable getMove(int keyCode)
```

Provides the `HNavigable` object that is navigated to when a particular key is pressed.

**Specified By:**

`getMove(int)` in interface `HNavigable`

**Parameters:**

`keyCode` - The key code of the pressed key.

**Returns:**

Returns the `HNavigable` object, or if no `HNavigable` is associated with the `keyCode` then returns null.

---

**getNavigationKeys()**

```
public int[] getNavigationKeys()
```

Retrieve the set of key codes which this component maps to navigation targets.

**Specified By:**

```
getNavigationKeys() in interface HNavigationInputPreferred
```

**Returns:**

an array of key codes, or `null` if no navigation targets are set on this component.

---

**getTextContent(int)**

```
public java.lang.String getTextContent(int state)
```

Gets the text content used in this `HSinglelineEntry`.

Note that `HSinglelineEntry` components do not support separate pieces of textual content per state (as defined in `HState`) --- rather a single piece of content is defined for all its interaction states.

**Overrides:**

```
getTextContent(int) in class HVisible
```

**Parameters:**

`state` - The state of the component for which this content should be displayed. This parameter shall be ignored.

**Returns:**

the text content used in this `HSinglelineEntry`

**See Also:**

```
setTextContent(String, int)
```

---

**getType()**

```
public int getType()
```

Retrieve the desired input type for this component. This value should be set to indicate to the system which input keys are required by this component.

**Specified By:**

```
getType() in interface HKeyboardInputPreferred
```

**Returns:**

one of `INPUT_ALPHANUMERIC`, `INPUT_NUMERIC`, `INPUT_ALPHA`, or `INPUT_CUSTOMIZED`.

---

**getValidInput()**

```
public char[] getValidInput()
```

Retrieve the customized input character range. The return value of this method should reflect the range of input keys which the component wishes to see, should `getType()` return `INPUT_CUSTOMIZED`. This method may return `null` if and only if it can guarantee that customized input is *never* required.

If customized input can be used but no specific customized input is set, this method should return the same range as the `INPUT_ALPHANUMERIC` input type, i.e. that range of characters for which the `java.lang.Character.isLetterOrDigit` returns `true`.

**Specified By:**

```
getValidInput() in interface HKeyboardInputPreferred
```

**Returns:**

an array containing the characters which this component expects the platform to provide, or `null` to indicate that customized characters are not used.

---

**insertChar(char)**

```
public boolean insertChar(char c)
```

Insert a character at the current caret position, subject to the maximum number of input characters.

**Parameters:**

`c` - the character to insert

**Returns:**

true if the character was inserted, false otherwise.

---

**isSelected()**

```
public boolean isSelected()
```

Indicates if this component has focus.

**Specified By:**

`isSelected()` in interface `HNavigable`

**Returns:**

true if the component has focus, otherwise returns false.

---

**processHFocusEvent(HFocusEvent)**

```
public void processHFocusEvent(HFocusEvent evt)
```

Process an `HFocusEvent` sent to this `HSinglelineEntry`.

**Specified By:**

`processHFocusEvent(HFocusEvent)` in interface `HNavigationInputPreferred`

**Parameters:**

`evt` - the `HFocusEvent` to process.

---

**processHKeyEvent(HKeyEvent)**

```
public void processHKeyEvent(HKeyEvent evt)
```

Process an `HKeyEvent` sent to this `HSinglelineEntry`.

**Specified By:**

`processHKeyEvent(HKeyEvent)` in interface `HKeyboardInputPreferred`

**Parameters:**

`evt` - the `HKeyEvent` to process.

---

**processHTextEvent(HTextEvent)**

```
public void processHTextEvent(HTextEvent evt)
```

Process an `HTextEvent` sent to this `HSinglelineEntry`.

**Specified By:**

`processHTextEvent(HTextEvent)` in interface `HKeyboardInputPreferred`

**Parameters:**

`evt` - the `HTextEvent` to process.

---

**setCaretCharPosition(int)**

```
public int setCaretCharPosition(int position)
```

Sets the position of the text insertion caret for this text component. If position is not valid for the current content the caret is moved to the nearest position.

**Parameters:**

`position` - the new position of the text insertion caret.

**Returns:**

the new caret position.

---

**setDefaultLook(HSinglelineEntryLook)**

```
public static void setDefaultLook(HSinglelineEntryLook look)
```

Sets the default `HLook` for further `HSinglelineEntry` Components.

**Parameters:**

`look` - The `HLook` that will be used by default when creating a new `HSinglelineEntry` component. Note that this parameter may be null, in which case newly created components shall not draw themselves until a non-null look is set using the `setLook(HLook)` method.

---

**setEchoChar(char)**

```
public void setEchoChar(char c)
```

Sets the number of character to echo for this component.

**Parameters:**

`c` - the character used to echo any input, e.g. if `c == '*'` a password-style input will be displayed. If `c` is zero (ASCII NUL), then all characters will be echoed on-screen, this is the default behavior.

---

**setEditMode(boolean)**

```
public void setEditMode(boolean edit)
```

Set the editing mode for this `HSinglelineEntry`.

This method is provided for the convenience of component implementors. Interoperable applications shall not call this method. It cannot be made protected because interfaces cannot have protected methods.

**Specified By:**

`setEditMode(boolean)` in interface `HKeyboardInputPreferred`

**Parameters:**

`edit` - true to switch this component into edit mode, false otherwise.

**See Also:**

`getEditMode()`

---

**setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)**

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
    HNavigable right)
```

Set the focus control for an `HNavigable` component. Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is a convenience function for application programmers where a standard up, down, left and right focus traversal between components is required.

Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

Note that this API does not prevent the creation of "isolated" `HNavigable` components --- authors should endeavor to avoid confusing the user.

**Specified By:**

```
setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) in
interface HNavigable
```

**Parameters:**

`up` - The `HNavigable` component to move to, when the user generates a `VK_UP` `KeyEvent`. If there is no `HNavigable` component to move "up" to, then null should be specified.

`down` - The `HNavigable` component to move to, when the user generates a `VK_DOWN` `KeyEvent`. If there is no `HNavigable` component to move "down" to, then null should be specified.

`left` - The `HNavigable` component to move to, when the user generates a `VK_LEFT` `KeyEvent`. If there is no `HNavigable` component to move "left" to, then null should be specified.

`right` - The `HNavigable` component to move to, when the user generates a `VK_RIGHT` `KeyEvent`. If there is no `HNavigable` component to move "right" to, then null should be specified.

### setGainFocusSound(HSound)

```
public void setGainFocusSound(HSound sound)
```

Associate a sound with gaining focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_GAINED`. This sound will start to be played when an object implementing this interface gains focus. It is not guaranteed to be played to completion. If the object implementing this interface loses focus before the audio completes playing, the audio will be truncated. Applications wishing to ensure the audio is always played to completion must implement special logic to slow down the focus transitions.

By default, an `HNavigable` object does not have any gain focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

**Specified By:**

```
setGainFocusSound(HSound) in interface HNavigable
```

**Parameters:**

`sound` - the sound to be played, when the component gains focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

### setLook(HLook)

```
public void setLook(HLook hlook)
```

Sets the `HLook` for this component.

**Overrides:**

```
setLook(HLook) in class HVisible
```

**Parameters:**

`hlook` - The `HLook` that is to be used for this component. Note that this parameter may be null, in which case the component will not draw itself until a look is set.

**Throws:**

`HInvalidLookException` - If the `Look` is not an `HSinglelineEntryLook`.



---

**setLoseFocusSound(HSound)**

```
public void setLoseFocusSound(HSound sound)
```

Associate a sound with losing focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_LOST`. This sound will start to be played when an object implementing this interface loses focus. It is not guaranteed to be played to completion. It is implementation dependent whether and when this sound will be truncated by any gain focus sound played by the next object to gain focus.

By default, an `HNavigable` object does not have any lose focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

**Specified By:**

`setLoseFocusSound(HSound)` in interface `HNavigable`

**Parameters:**

`sound` - the sound to be played, when the component loses focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

---

**setMaxCharsPerLine(int)**

```
public void setMaxCharsPerLine(int maxCol)
```

Set maximum number of characters per (single) line.

**Parameters:**

`maxCol` - the maximum number of characters per line.

---

**setMove(int, HNavigable)**

```
public void setMove(int keyCode, HNavigable target)
```

Defines the navigation path from the current `HNavigable` to another `HNavigable` when a particular key is pressed.

Note that `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

**Specified By:**

`setMove(int, HNavigable)` in interface `HNavigable`

**Parameters:**

`keyCode` - The key code of the pressed key. Any numerical keycode is allowed, but the platform may not be able to generate all keycodes. Application authors should only use keys for which `HRcCapabilities.isSupported()` or `HKeyCapabilities.isSupported()` returns true.

`target` - The target `HNavigable` object that should be navigated to. If a target is to be removed from a particular navigation path, then `null` should be specified.

---

**setTextContent(String, int)**

```
public void setTextContent(java.lang.String string, int state)
```

Sets the text content used in this `HSinglelineEntry`.

Note that `HSinglelineEntry` components do not support separate pieces of textual content per state (as defined in `HState`) --- rather a single piece of content is defined for all its interaction states.

Additionally, the `setTextContent(String, int)` method truncates the string according to the current `maxChars` setting.

**Overrides:**

`setTextContent(String, int)` in class `HVisible`

**Parameters:**

`string` - The content. If the content is null, then any currently assigned content shall be removed for the specified state.

`state` - The state of the component for which this content should be displayed. This parameter shall be ignored and considered to have the value `ALL_STATES`.

**See Also:**

`getTextContent(int)`

---

**setType(int)**

```
public void setType(int type)
```

Set the type of permitted keyboard entry.

**Parameters:**

`type` - one of `INPUT_ANY`, `INPUT_ALPHANUMERIC`, `INPUT_ALPHA`, `INPUT_NUMERIC` or `INPUT_CUSTOMIZED`

---

**setValidInput(char[])**

```
public void setValidInput(char[] inputChars)
```

Defines the set of the characters which are valid for customized keyboard input, i.e. when the input type is set to `INPUT_CUSTOMIZED`.

**Parameters:**

`inputChars` - an array of characters which comprises the valid input characters.

# org.havi.ui HSinglelineEntryLook

## Syntax

```
public class HSinglelineEntryLook implements HLook
```

```
java.lang.Object
|
+--org.havi.ui.HSinglelineEntryLook
```

## Direct Known Subclasses:

```
HMultilineEntryLook
```

## All Implemented Interfaces:

```
java.lang.Cloneable, HLook
```

## Description

The `HSinglelineEntryLook` class is used by the `HSinglelineEntry` component to display the entering of text. This look will be provided by the platform and the exact way in which it is rendered will be platform dependant.

The `HSinglelineEntryLook` class draws the content set on an `HSinglelineEntry`. It uses the `getTextContent(int)` method to determine the content to render. The interaction state of the `HSinglelineEntry` is ignored.

This is the default look that is used by `HSinglelineEntry` and its subclasses. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
-----------	-------------	---------------	------------	------------

None.

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

None.

## See Also:

```
HSinglelineEntry
```

## Constructors

### HSinglelineEntryLook()

```
public HSinglelineEntryLook()
```

Creates a `HSinglelineEntryLook` object. See the class description for details of constructor parameters and default values.

## Methods

---

### `getInsets(HVisible)`

```
public java.awt.Insets getInsets(HVisible visible)
```

Determines the insets of this `HLook`, which indicate the size of the border. This area is reserved for the `HLook` to use for drawing borders around the associated `HVisible`.

**Specified By:**

`getInsets(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

the insets of this `HLook`.

---

### `getMaximumSize(HVisible)`

```
public java.awt.Dimension getMaximumSize(HVisible hvisible)
```

Gets the maximum size of the `HVisible` component when drawn with this `HLook`.

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook`) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If the `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) then the return value is the current size of the `HVisible` (as returned by `HVisible#getSize`).
2. If the `HLook` does not support scaling of content or no scaling is requested, and content is set then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If there is no content set then a maximum size of `[ Short.MAX_VALUE, Short.MAX_VALUE ]` is returned as a `Dimension`.

**Specified By:**

`getMaximumSize(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating this `HLook` maximum size.

**See Also:**

`getMaximumSize()`

---

### `getMinimumSize(HVisible)`

```
public java.awt.Dimension getMinimumSize(HVisible hvisible)
```

Gets the minimum size of the `HVisible` component when drawn with this `HLook`.

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook`) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If the `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) and content is set then the return value is the size of the smallest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
2. If the `HLook` does not support scaling of content or no scaling is requested, *and* content is set then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If no content is available but a default preferred size has been set using `setDefaultSize(Dimension)` has been called to set then the return value is this value (as obtained with `getDefaultSize()`) plus any additional dimensions that the `HLook` requires for border decoration etc.
4. If there is no content or default size set then the return value is an implementation-specific minimum size plus any additional dimensions that the `HLook` requires for border decoration etc.

**Specified By:**

`getMinimumSize(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating this `HLook` minimum size.

**See Also:**

`getMinimumSize()`

### **getPreferredSize(HVisible)**

```
public java.awt.Dimension getPreferredSize(HVisible hvisible)
```

Gets the preferred size of the `HVisible` component when drawn with this `HLook`.

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook`) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If a default preferred size has been set for this `HVisible` (using `setDefaultSize(Dimension)`) then the return value is this size (obtained with `getDefaultSize()`) plus any additional dimensions that the `HLook` requires for border decoration etc.
2. If this `HLook` does not support scaling of content or no scaling is requested, and content is present then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If this `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) and content is set then the return value is the current size of the `HVisible` as returned by `getSize()`.
4. If there is no content and no default size set then the return value is the current size of the `HVisible` as returned by `getSize()`.

**Specified By:**

`getPreferredSize(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating the preferred size of the `HVisible` when drawn with this `HLook`.

**See Also:**

`getPreferredSize()`, `setDefaultSize(Dimension)`

**isOpaque(HVisible)**

```
public boolean isOpaque(HVisible visible)
```

Returns true if the entire painted area of the `HVisible` when using this look is fully opaque, i.e. the `showLook(Graphics, HVisible, int)` method guarantees that all pixels are painted in an opaque `Color`.

The default value is implementation specific and depends on the background painting mode of the given `HVisible`. The consequences of an invalid overridden value are implementation specific.

**Specified By:**

`isOpaque(HVisible)` in interface `HLook`

**Parameters:**

`visible` - the visible to test

**Returns:**

true if all the pixels with the `java.awt.Component#getBounds` method of an `HVisible` using this look are fully opaque, i.e. the `showLook(Graphics, HVisible, int)` method guarantees that all pixels are painted in an opaque `Color`, otherwise false.

**showLook(Graphics, HVisible, int)**

```
public void showLook(java.awt.Graphics g, HVisible visible, int state)
```

The `showLook(Graphics, HVisible, int)` method is responsible for repainting the entire `HVisible` component, (including any content set on the component, and the component background), subject to the `clipRect` of the `Graphics` object passed to it.

The `showLook(Graphics, HVisible, int)` method should not modify the `clipRect` of the `Graphics` object that is passed to it.

For looks which draw content (e.g. `HTextLook`, `HGraphicLook` and `HAnimateLook`), if no content is associated with the component, the `showLook(Graphics, HVisible, int)` method paints the component with its current background `Color` according to the `setBackgroundMode(int)` method of `HVisible` and draws any (implementation-specific) borders. Note that by default the background mode is set so as to **not** paint a background. Furthermore on platforms which support transparent colors the background `Color` may be partially or completely transparent.

Any resources **explicitly** associated with an `HLook` should be loaded by the `HLook` during its creation, etc., or via its `setXXX()` methods. Note that the "standard" looks don't load content by default.

This method is called from the `paint(Graphics)` method of `HVisible` and must never be called from elsewhere. Components wishing to redraw themselves should call their repaint method in the usual way.

**Specified By:**

`showLook(Graphics, HVisible, int)` in interface `HLook`

**Parameters:**

`g` - the graphics context.

`visible` - the visible.

`state` - the state parameter indicates the state of the visible, allowing the look to render the appropriate content for that state. Note that some components (e.g. `HStaticRange`, `HRange`, `HRangeValue`) do not use state-based content).

---

**widgetChanged(HVisible, HChangeData[])**

```
public void widgetChanged(HVisible visible, HChangeData[] changes)
```

Called by the `HVisible` whenever its content, state, or any other data changes. See the class description of `HVisible` for more information about the `changes` parameter.

The implementation of this method should work out which graphical areas of the `HVisible` have changed and make any relevant calls to trigger the repainting of those areas.

A minimum implementation of this method could simply call

```
visible.repaint()
```

**Specified By:**

```
widgetChanged(HVisible, HChangeData[] ) in interface HLook
```

**Parameters:**

`visible` - the `HVisible` which has changed

`changes` - an array containing hint data and associated hint objects. If this argument is `null` a full repaint will be triggered.

# org.havi.ui

# HSound

## Syntax

```
public class HSound
    java.lang.Object
    |
    +--org.havi.ui.HSound
```

## Description

The `HSound` class is used to represent an audio clip. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
-----------	-------------	---------------	------------	------------

None.

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

The starting position of any audio clip to be played.	At the beginning of the audio clip.	---	---
---	-------------------------------------	-----	-----

## Constructors

### HSound()

```
public HSound()
```

Creates an `HSound` object. See the class description for details of constructor parameters and default values.

## Methods

### dispose()

```
public void dispose()
```

If the `HSound` object is playing / looping then it will be stopped. The dispose method then discards all sample resources used by the `HSound` object. This mechanism resets the `HSound` object to the state before a `load()` method was invoked.

### load(String)

```
public void load(java.lang.String location)
```



Loads data synchronously into an `HSound` object from an audio sample in the specified file. If the object already contains data, this method shall perform the following sequence:

- stop the sample if it is playing or looping.
- dispose of the old data and any associated resources, as if the `dispose()` method had been called.
- load the new data synchronously.

**Parameters:**

`location` - the name of a file containing audio data in a recognized file format.

**Throws:**

`java.io.IOException` - if the sample cannot be loaded due to an IO problem.

`java.lang.SecurityException` - if the caller does not have sufficient rights to access the specified audio sample.

---

### load(URL)

```
public void load(java.net.URL contents)
```

Loads data synchronously into an `HSound` object from an audio sample indicated by a URL. If the object already contains data, this method shall perform the following sequence:

- stop the sample if it is playing or looping.
- dispose of the old data and any associated resources, as if the `dispose()` method had been called.
- load the new data synchronously.

**Parameters:**

`contents` - a URL referring to the data to load.

**Throws:**

`java.io.IOException` - if the audio sample cannot be loaded due to an IO problem.

`java.lang.SecurityException` - if the caller does not have sufficient rights to access the specified audio sample.

---

### loop()

```
public void loop()
```

Starts the `HSound` class looping from the beginning of its associated audio data. If the sample data has not been completely loaded, this method has no effect.

When the audio data has been played in its entirety, then it should be played again from the beginning of its associated data, so as to cause a "seamless" continuous (infinite) audio playback - until the next stop, or play method is invoked. Note that the audio data is played back asynchronously, there is no mechanism for synchronization with other classes presenting sounds, images, or video.

This method may fail "silently" if (local) audio facilities are unavailable on the platform.

---

### play()

```
public void play()
```

Starts the `HSound` class playing from the beginning of its associated audio data. If the sample data has not been completely loaded, this method has no effect.

When the audio data has been played in its entirety then no further audible output should be made until the next play or loop method is invoked. Note that the audio data is played back asynchronously. There is no mechanism for synchronization with other classes presenting sounds, images, or video.

This method may fail "silently" if (local) audio facilities are unavailable on the platform.

---

**set(byte[])**

```
public void set(byte[] data)
```

Constructs an `HSound` object from an array of bytes encoded in the same encoding format as when reading this type of audio sample data from a file. If the object already contains data, this method shall perform the following sequence:

- stop the sample if it is playing or looping.
- dispose of the old data and any associated resources, as if the `dispose()` method had been called.
- load the new data synchronously.

If the byte array does not contain a valid audio sample then this method shall throw a `java.lang.IllegalArgumentException`.

**Parameters:**

`data` - the data for the `HSound` object encoded in the specified format for audio sample files of this type.

---

**stop()**

```
public void stop()
```

Stops the `HSound` class playing its associated audio data.

Note that, if a play or loop method is invoked, after a stop, then presentation of the audio data will restart from the beginning of the audio data, rather than from the position where the audio data was stopped.

# org.havi.ui

## HState

### Syntax

```
public interface HState
```

### All Known Implementing Classes:

```
HVisible
```

### Description

The `HState` interface encapsulates constants for component states which are used in the various `HVisible` `setContent` and `getContent` methods, to indicate which state the specified content is to be set.

There are two sets of constants defined in this interface. The first set are mutually exclusive state bits, which define properties of the component. The order of the states is important; each state has precedence over the one before it when considering the effect on the component. For example, the `DISABLED_STATE_BIT` property is considered the most significant property of a state. The state bits are shown in the table below.

Name	Interpretation
<code>FOCUSED_STATE_BIT</code>	component has focus
<code>ACTIONED_STATE_BIT</code>	component is actioned
<code>DISABLED_STATE_BIT</code>	component is disabled

The state bits combine to provide the actual interaction states which `HVisible` components may be in. Content for components is set on these states, not the state bits shown above. The interaction states are shown in the table below.

State	DISABLED_BIT	ACTIONED_BIT	FOCUSED_BIT
<code>NORMAL_STATE</code>	0	0	0
<code>FOCUSED_STATE</code>	0	0	1
<code>ACTIONED_STATE</code>	0	1	0
<code>ACTIONED_FOCUSED_STATE</code>	0	1	1
<code>DISABLED_STATE</code>	1	0	0
<code>DISABLED_FOCUSED_STATE</code>	1	0	1

<code>DISABLED_ACTIONED_STATE</code>	1	1	0
<code>DISABLED_ACTIONED_FOCUSED_STATE</code>	1	1	1

**See Also:**

`setTextContent(String, int)`, `getTextContent(int)`, `setGraphicContent(Image, int)`, `getGraphicContent(int)`, `setAnimateContent(Image[], int)`, `getAnimateContent(int)`, `setContent(Object, int)`, `getContent(int)`, `setInteractionState(int)`, `getInteractionState()`

**Fields****ACTIONED\_FOCUSED\_STATE**

```
public static final int ACTIONED_FOCUSED_STATE
```

This state indicates that the widget has been actioned, and has focus. `HSwitchable` components may stay in this state until they are actioned again. This state is applicable to all `HActionable` and `HSwitchable` components.

**See Also:**

`HActionable`, `HSwitchable`

**ACTIONED\_STATE**

```
public static final int ACTIONED_STATE
```

This state indicates that the widget has been actioned, but does not have focus. `HSwitchable` components may stay in this state until they are actioned again. This state is applicable to all `HActionable` and `HSwitchable` components.

**See Also:**

`HActionable`, `HSwitchable`

**ACTIONED\_STATE\_BIT**

```
public static final int ACTIONED_STATE_BIT
```

This state bit indicates that the widget has been actioned. `HActionable` components only have this bit set for the duration of the calls to their registered `ActionListeners`, whereas `HSwitchable` components may remain with the `ACTIONED` bit set until further user input causes them to leave it. If state-based content is not used, the associated look should visually distinguish components with this bit set e.g. by drawing them as "pushed in".

**ALL\_STATES**

```
public static final int ALL_STATES
```

Constant used to indicate all of the applicable states for a given component.

Note that the `ALL_STATES` constant should only be used in setting content `setTextContent(String, int)` `setGraphicContent(Image, int)` `setAnimateContent(Image[], int)` `setContent(Object, int)`

The ALL\_STATES constant should not be used for retrieving content: `getTextContent(int)`  
`getGraphicContent(int)` `getAnimateContent(int)` `getContent(int)`

---

### DISABLED\_ACTIONED\_FOCUSED\_STATE

```
public static final int DISABLED_ACTIONED_FOCUSED_STATE
```

This state indicates that the widget has been actioned and has input focus but is disabled. This state is applicable to all `HSwitchable` components.

**See Also:**

`HSwitchable`

---

### DISABLED\_ACTIONED\_STATE

```
public static final int DISABLED_ACTIONED_STATE
```

This state indicates that the widget has been actioned but is disabled. This state is applicable to all `HSwitchable` components.

**See Also:**

`HSwitchable`

---

### DISABLED\_FOCUSED\_STATE

```
public static final int DISABLED_FOCUSED_STATE
```

This state indicates that the widget has input focus but is disabled. This state is applicable to all `HNavigable` components.

**See Also:**

`HNavigable`

---

### DISABLED\_STATE

```
public static final int DISABLED_STATE
```

This state indicates that the widget is disabled. This state is applicable to all `HVisible` components.

**See Also:**

`HVisible`

---

### DISABLED\_STATE\_BIT

```
public static final int DISABLED_STATE_BIT
```

This state bit indicates that the component is disabled. Regardless of other bits being set it shall not respond to user action, shortcut keys or mouse clicks. However the component is only disabled at the HAVi level. It may still receive AWT events directly, which interoperable components must ignore. If state-based content is not used the associated look should visually distinguish components with this bit set e.g. by graying them out.

---

### FIRST\_STATE

```
public static final int FIRST_STATE
```

Constant used to indicate the value of the first (builtin) component state.

---

**FOCUSED\_STATE**

```
public static final int FOCUSED_STATE
```

This state indicates that the widget has input focus. This state is applicable to all `HNavigable` components.

**See Also:**

`HNavigable`

---

**FOCUSED\_STATE\_BIT**

```
public static final int FOCUSED_STATE_BIT
```

This state bit indicates that the widget has the input focus. This state is only valid for widgets implementing `HNavigable`. If state-based content is not used, the associated look should visually distinguish components with this bit set e.g. by highlighting them.

---

**LAST\_STATE**

```
public static final int LAST_STATE
```

Constant used to indicate the value of the last (builtin) component state.

---

**NORMAL\_STATE**

```
public static final int NORMAL_STATE
```

This constant (i.e. no state bits set) indicates that the widget is in its normal state. This state is applicable to all `HVisible` components.

**See Also:**

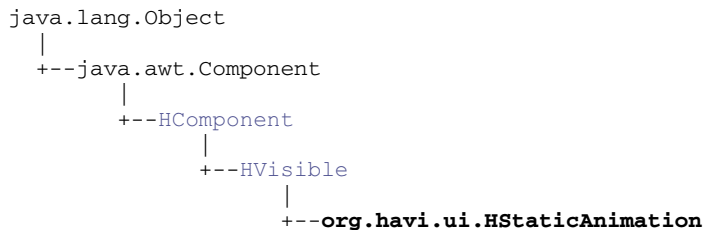
`HVisible`

# org.havi.ui

## HStaticAnimation

### Syntax

public class HStaticAnimation extends HVisible implements HNoInputPreferred, HAnimateEffect



### Direct Known Subclasses:

[HAnimation](#)

### All Implemented Interfaces:

[HAnimateEffect](#), [HMatteLayer](#), [HNoInputPreferred](#), [HState](#),  
[java.awt.image.ImageObserver](#), [java.awt.MenuContainer](#), [java.io.Serializable](#), [TestOpacity](#)

### Description

The [HStaticAnimation](#) is a user interface component used to display animated graphical content but which does *not* permit the user to navigate (focus) upon it. By default it uses the [HAnimateLook](#) to render itself.

The [HStaticAnimation](#) class supports animating images a finite number of times or infinitely (continuously), and either forward or in alternating directions with a specified time delay between the rendering of consecutive images. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
height	height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

images-Normal	The array of images to be used as the content for the <code>NORMAL_STATE</code> state of this component.	null	<code>setAnimateContent (Image[] , int)</code>	<code>getAnimateContent (int)</code>
delay	The delay between the presentation of successive content in the animation, in units of 0.1 second	1 (i.e. 0.1 seconds)	<code>setDelay (int)</code>	<code>getDelay ()</code>
repeat-Count	The number of times that the animation is to be played.	<code>REPEAT_INFINITE</code>	<code>setRepeatCount (int)</code>	<code>getRepeatCount ()</code>
playMode	The playing mode for the animation.	<code>PLAY_REPEATING</code>	<code>setPlayMode (int)</code>	<code>getPlayMode ()</code>

#### Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte ( <code>HMatte</code> ).	none (i.e. <code>getMatte()</code> returns null)	<code>setMatte (HMatte)</code>	<code>getMatte ()</code>
The text layout manager responsible for text formatting.	An <code>HDefaultTextLayoutManager</code> object.	<code>setTextLayoutManager (HTextLayoutManager)</code>	<code>getTextLayoutManager ()</code>
The background painting mode	<code>NO_BACKGROUND_FILL</code>	<code>setBackgroundMode (int)</code>	<code>getBackgroundMode ()</code>
The default preferred size	not set (i.e. null) unless specified by width and height parameters	<code>setDefaultSize (Dimension)</code>	<code>getDefaultSize ()</code>
The horizontal content alignment	<code>HALIGN_CENTER</code>	<code>setHorizontalAlignment (int)</code>	<code>getHorizontalAlignment ()</code>
The vertical content alignment	<code>VALIGN_CENTER</code>	<code>setVerticalAlignment (int)</code>	<code>getVerticalAlignment ()</code>
The content scaling mode	<code>RESIZE_NONE</code>	<code>setResizeMode (int)</code>	<code>getResizeMode ()</code>
The initial piece of content to be presented, i.e. its position in the content array.	0	<code>setPosition (int)</code>	<code>getPosition ()</code>
By default the animation should be stopped. Hence, to start the animation its start method must be explicitly invoked. This mechanism allows for animations that are programmatically controlled, e.g. via the <code>setPosition</code> method.	"stopped"	<code>start () / stop ()</code>	<code>isAnimated ()</code>



The default "look" for this class.	A platform specific <code>HAnimateLook</code>	<code>setDefaultLook(HAnimateLook)</code>	<code>getDefaultLook()</code>
The "look" for this object.	The <code>HAnimateLook</code> returned from <code>HStaticAnimation.getDefaultLook</code> when this object was created.	<code>setLook(HLook)</code>	<code>getLook()</code>

## Constructors

---

### `HStaticAnimation()`

```
public HStaticAnimation()
```

Creates an `HStaticAnimation` object. See the class description for details of constructor parameters and default values.

---

### `HStaticAnimation(Image[], int, int, int)`

```
public HStaticAnimation(java.awt.Image[] imagesNormal, int delay, int playMode,
                        int repeatCount)
```

Creates an `HStaticAnimation` object. See the class description for details of constructor parameters and default values.

---

### `HStaticAnimation(Image[], int, int, int, int, int, int, int)`

```
public HStaticAnimation(java.awt.Image[] imagesNormal, int delay, int playMode,
                        int repeatCount, int x, int y, int width, int height)
```

Creates an `HStaticAnimation` object. See the class description for details of constructor parameters and default values.

## Methods

---

### `getDefaultLook()`

```
public static HAnimateLook getDefaultLook()
```

Returns the currently set default `HLook` for `HStaticAnimation` components.

**Returns:**

The `HLook` that is used by default when creating a new `HStaticAnimation` component.

---

### `getDelay()`

```
public int getDelay()
```

Gets the presentation delay for this `HStaticAnimation`.

**Specified By:**

`getDelay()` in interface `HAnimateEffect`

**Returns:**

the presentation delay in units of 0.1 seconds.

---

**getPlayMode()**

```
public int getPlayMode()
```

Gets the playing mode for this `HStaticAnimation`.

**Specified By:**

`getPlayMode()` in interface `HAnimateEffect`

**Returns:**

the play mode for this `HStaticAnimation`.

---

**getPosition()**

```
public int getPosition()
```

Get the current index into the content array which this `HStaticAnimation` is using to display content.

**Specified By:**

`getPosition()` in interface `HAnimateEffect`

**Returns:**

the index of the content currently being displayed, in the range  $0 \leq \text{index} < \text{length}$

---

**getRepeatCount()**

```
public int getRepeatCount()
```

Gets the number of times that this `HStaticAnimation` is to be played. Note that this method does *not* return the number of repeats that are remaining to be played.

**Specified By:**

`getRepeatCount()` in interface `HAnimateEffect`

**Returns:**

the total number of times that an `HStaticAnimation` is to be played. The returned value shall be greater than zero, or `REPEAT_INFINITE`.

---

**isAnimated()**

```
public boolean isAnimated()
```

This method indicates the animation (running) state of the `HStaticAnimation`.

**Specified By:**

`isAnimated()` in interface `HAnimateEffect`

**Returns:**

`true` if this `HStaticAnimation` is running, i.e. the `start` method has been invoked - `false` otherwise.

---

**setDefaultLook(HAnimateLook)**

```
public static void setDefaultLook(HAnimateLook hlook)
```

Sets the default `HLook` for further `HStaticAnimation` Components.

---

**Parameters:**

`hlook` - The `HLook` that will be used by default when creating a new `HStaticAnimation` component. Note that this parameter may be null, in which case newly created components shall not draw themselves until a non-null look is set using the `setLook(HLook)` method.

---

**setDelay(int)**

```
public void setDelay(int count)
```

Sets the delay between the presentation of successive pieces of content (frames).

After calling `setDelay(int)` on a currently playing `HStaticAnimation`, there is no guarantee that one or more frames will not be displayed using the previous delay until the new delay value takes effect.

**Specified By:**

`setDelay(int)` in interface `HAnimateEffect`

**Parameters:**

`count` - the content presentation delay in units of 0.1 seconds duration. If `count` is less than one "unit", then it shall be treated as if it were a delay of one "unit", i.e. 0.1 seconds.

---

**setLook(HLook)**

```
public void setLook(HLook hlook)
```

Sets the `HLook` for this component.

**Overrides:**

`setLook(HLook)` in class `HVisible`

**Parameters:**

`hlook` - The `HLook` that is to be used for this component. Note that this parameter may be null, in which case the component will not draw itself until a look is set.

**Throws:**

`HInvalidLookException` - If the `HLook` is not an `HAnimateLook`.

---

**setPlayMode(int)**

```
public void setPlayMode(int mode)
```

Sets the playing mode for this `HStaticAnimation`. If the animation is already running a call to `setPlayMode` will change the current value and affect the animation immediately. The position of the animation is unchanged.

**Specified By:**

`setPlayMode(int)` in interface `HAnimateEffect`

**Parameters:**

`mode` - the play mode for this `HStaticAnimation`, which must be either `PLAY_ALTERNATING` or `PLAY_REPEATING`.

---

**setPosition(int)**

```
public void setPosition(int position)
```

Set this `HStaticAnimation` to display the content at the specified position. If the animation is already running a call to `setPosition` will change the current value and affect the animation immediately.

**Specified By:**

`setPosition(int)` in interface `HAnimateEffect`

**Parameters:**

`position` - an index into the content array which specifies the next piece of content to be displayed. If `position` is less than 0, then the array element at index 0 is displayed, if `position` is greater than or equal to the length of the content array, then the array element at index `[length-1]` will be used.

---

**setRepeatCount(int)**

```
public void setRepeatCount(int count)
```

Sets the number of times that this `HStaticAnimation` should be played. If the animation is already running a call to `setRepeatCount` will change the current value and reset the current number of repeats to 0, affecting the animation immediately.

**Specified By:**

`setRepeatCount(int)` in interface `HAnimateEffect`

**Parameters:**

`count` - the number of times that an `HStaticAnimation` should be played. Valid values of the repeat count are one or more, and `REPEAT_INFINITE`.

---

**start()**

```
public void start()
```

This method starts this `HStaticAnimation` playing. If `start` is called when the animation is already running it resets the animation according to the current play mode, as returned by `getPlayMode()`.

**Specified By:**

`start()` in interface `HAnimateEffect`

---

**stop()**

```
public void stop()
```

This method indicates that the running `HStaticAnimation` should be stopped. After calling this method, there is no guarantee that one or more frames will not be displayed before the animation actually stops playing. If the animation is already stopped further calls to `stop` have no effect.

**Specified By:**

`stop()` in interface `HAnimateEffect`

# org.havi.ui

## HStaticIcon

### Syntax

public class HStaticIcon extends HVisible implements HNoInputPreferred

```

java.lang.Object
|
+--java.awt.Component
    |
    +--HComponent
        |
        +--HVisible
            |
            +--org.havi.ui.HStaticIcon
  
```

### Direct Known Subclasses:

[HIcon](#)

### All Implemented Interfaces:

[HMatteLayer](#), [HNoInputPreferred](#), [HState](#), [java.awt.image.ImageObserver](#), [java.awt.MenuContainer](#), [java.io.Serializable](#), [TestOpacity](#)

### Description

The [HStaticIcon](#) is a user interface component used to display static graphical content which does *not* permit the user to navigate (focus) upon it. By default it uses the [HGraphicLook](#) to render itself. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
height	height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
imageNormal	The image to be used as the content for the <code>NORMAL_STATE</code> state of this component.	null	setGraphicContent(Image, int)	getGraphicContent(int)

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte ( <code>HMatte</code> ).	none (i.e. <code>getMatte()</code> returns null)	<code>setMatte(HMatte)</code>	<code>getMatte()</code>
The text layout manager responsible for text formatting.	An <code>HDefaultTextLayoutManager</code> object.	<code>setTextLayoutManager(HTextLayoutManager)</code>	<code>getTextLayoutManager()</code>
The background painting mode	<code>NO_BACKGROUND_FILL</code>	<code>setBackgroundMode(int)</code>	<code>setBackgroundMode()</code>
The default preferred size	not set (i.e. null) unless specified by width and height parameters	<code>setDefaultSize(Dimension)</code>	<code>getDefaultSize()</code>
The horizontal content alignment	<code>HALIGN_CENTER</code>	<code>setHorizontalAlignment(int)</code>	<code>getHorizontalAlignment()</code>
The vertical content alignment	<code>VALIGN_CENTER</code>	<code>setVerticalAlignment(int)</code>	<code>getVerticalAlignment()</code>
The content scaling mode	<code>RESIZE_NONE</code>	<code>setResizeMode(int)</code>	<code>getResizeMode()</code>
The default "look" for this class.	A platform specific <code>HGraphicLook</code>	<code>setDefaultLook(HGraphicLook)</code>	<code>getDefaultLook()</code>
The "look" for this object.	The <code>HGraphicLook</code> returned from <code>HStaticIcon.getDefaultLook</code> when this object was created.	<code>setLook(HLook)</code>	<code>getLook()</code>

## Constructors

---

### **HStaticIcon()**

```
public HStaticIcon()
```

Creates an `HStaticIcon` object. See the class description for details of constructor parameters and default values.

---

### **HStaticIcon(Image)**

```
public HStaticIcon(java.awt.Image imageNormal)
```

Creates an `HStaticIcon` object. See the class description for details of constructor parameters and default values.

---

**HStaticIcon(Image, int, int, int, int)**

```
public HStaticIcon(java.awt.Image imageNormal, int x, int y, int width, int height)
```

Creates an `HStaticIcon` object. See the class description for details of constructor parameters and default values.

## Methods

---

**getDefaultLook()**

```
public static HGraphicLook getDefaultLook()
```

Returns the currently set default `HLook` for `HStaticIcon` components.

**Returns:**

The `HLook` that is used by default when creating a new `HStaticIcon` component.

---

**setDefaultLook(HGraphicLook)**

```
public static void setDefaultLook(HGraphicLook hlook)
```

Sets the default `HLook` for further `HStaticIcon` Components.

**Parameters:**

`hlook` - The `HLook` that will be used by default when creating a new `HStaticIcon` component. Note that this parameter may be null, in which case newly created components shall not draw themselves until a non-null look is set using the `setLook(HLook)` method.

---

**setLook(HLook)**

```
public void setLook(HLook hlook)
```

Sets the `HLook` for this component.

**Overrides:**

`setLook(HLook)` in class `HVisible`

**Parameters:**

`hlook` - The `HLook` that is to be used for this component. Note that this parameter may be null, in which case the component will not draw itself until a look is set.

**Throws:**

`HInvalidLookException` - If the `HLook` is not an `HGraphicLook`.

# org.havi.ui HStaticRange

## Syntax

public class HStaticRange extends HVisible implements HNoInputPreferred, HOrientable

```

java.lang.Object
|
+--java.awt.Component
|   |
|   +--HComponent
|       |
|       +--HVisible
|           |
|           +--org.havi.ui.HStaticRange
  
```

## Direct Known Subclasses:

[HRange](#)

## All Implemented Interfaces:

[HMatteLayer](#), [HNoInputPreferred](#), [HOrientable](#), [HState](#), [java.awt.image.ImageObserver](#), [java.awt.MenuContainer](#), [java.io.Serializable](#), [TestOpacity](#)

## Description

The [HStaticRange](#) is a user interface component used to display a static value which is within a fixed range, but does *not* permit the user to navigate (focus) upon it. By default it uses the [HRangeLook](#) to render itself. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
height	height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
orientation	The "orientation" of the range object.	ORIENT_LEFT_T O_RIGHT	setOrientation( int)	getOrientation( )



minimum	The minimum value that can be returned by this range object.	0	<code>setRange(int, int)</code>	<code>getMinValue()</code>
maximum	The maximum value that can be returned by this range object.	100	<code>setRange(int, int)</code>	<code>getMaxValue()</code>
value	The current value returned by this range object.	0	<code>setValue(int)</code>	<code>getValue()</code>

#### Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte ( <code>HMatte</code> ).	none (i.e. <code>getMatte()</code> returns null)	<code>setMatte(HMatte)</code>	<code>getMatte()</code>
The text layout manager responsible for text formatting.	An <code>HDefaultTextLayoutManager</code> object.	<code>setTextLayoutManager(HTextLayoutManager)</code>	<code>getTextLayoutManager()</code>
The background painting mode	<code>NO_BACKGROUND_FILL</code>	<code>setBackgroundMode(int)</code>	<code>getBackgroundMode()</code>
The default preferred size	not set (i.e. null) unless specified by width and height parameters	<code>setDefaultSize(Dimension)</code>	<code>getDefaultSize()</code>
The horizontal content alignment	<code>HALIGN_CENTER</code>	<code>setHorizontalAlignment(int)</code>	<code>getHorizontalAlignment()</code>
The vertical content alignment	<code>VALIGN_CENTER</code>	<code>setVerticalAlignment(int)</code>	<code>getVerticalAlignment()</code>
The content scaling mode	<code>RESIZE_NONE</code>	<code>setResizeMode(int)</code>	<code>getResizeMode()</code>
The default "look" for this class.	A platform specific <code>HRangeLook</code>	<code>setDefaultLook(HRangeLook)</code>	<code>getDefaultLook()</code>
The "look" for this object.	The <code>HRangeLook</code> returned from <code>HStaticRange.getDefaultLook</code> when this object was created.	<code>setLook(HLook)</code>	<code>getLook()</code>
The offsets for the "thumb" of this range control	min = 0, max = 0	<code>setThumbOffsets(int, int)</code>	<code>getThumbMinOffset()</code> / <code>getThumbMaxOffset()</code>
The behavior of this range object with respect to its "thumb" values	<code>SLIDER_BEHAVIOR</code>	<code>setBehavior(int)</code>	<code>getBehavior()</code>

## Fields

---

### SCROLLBAR\_BEHAVIOR

```
public static final int SCROLLBAR_BEHAVIOR
```

The `HStaticRange` should behave as a scrollbar, i.e. the allowable values that may be set / returned for the `HStaticRange` should be affected by the "thumb" offsets, and hence its value should be able to vary between `[minimum + minThumbOffset, maximum - maxThumbOffset]`.

---

### SLIDER\_BEHAVIOR

```
public static final int SLIDER_BEHAVIOR
```

The `HStaticRange` should behave as a slider, i.e. the allowable values that may be set / returned for the `HStaticRange` should not be affected by the "thumb" offsets, and hence its value should be able to vary between `[minimum, maximum]`.

## Constructors

---

### `HStaticRange()`

```
public HStaticRange()
```

Creates an `HStaticRange` object. See the class description for details of constructor parameters and default values.

---

### `HStaticRange(int, int, int, int)`

```
public HStaticRange(int orientation, int minimum, int maximum, int value)
```

Creates an `HStaticRange` object. See the class description for details of constructor parameters and default values.

---

### `HStaticRange(int, int, int, int, int, int, int, int)`

```
public HStaticRange(int orientation, int minimum, int maximum, int value, int x, int y,
                    int width, int height)
```

Creates an `HStaticRange` object. See the class description for details of constructor parameters and default values.

## Methods

---

### `getBehavior()`

```
public int getBehavior()
```

Returns the behavior for this `HStaticRange`.

**Returns:**

the behavior for this `HStaticRange` .

---

**getDefaultLook()**

```
public static HRangeLook getDefaultLook()
```

Returns the currently set default `HLook` for `HStaticRange` components.

**Returns:**

The `HLook` that is used by default when creating a new `HStaticRange` component.

---

**getMaxValue()**

```
public int getMaxValue()
```

Get the maximum value of the range

**Returns:**

The maximum value of the range.

---

**getMinValue()**

```
public int getMinValue()
```

Gets the minimum of the range.

**Returns:**

The minimum value for the range

---

**getOrientation()**

```
public int getOrientation()
```

Retrieve the orientation of the `HStaticRange` . The orientation controls how an associated `HLook` lays out the component and affects the visual behavior of the `HAdjustmentEvent` and `HItemEvent` events. For example, the system might use this information to select appropriate key mappings for these events.

**Specified By:**

`getOrientation()` in interface `HOrientable`

**Returns:**

one of `ORIENT_LEFT_TO_RIGHT` , `ORIENT_RIGHT_TO_LEFT` , `ORIENT_TOP_TO_BOTTOM` , or `ORIENT_BOTTOM_TO_TOP` .

---

**getThumbMaxOffset()**

```
public int getThumbMaxOffset()
```

Returns the thumb offset for its maximum value.

**Returns:**

the thumb offset for its maximum value.

---

**getThumbMinOffset()**

```
public int getThumbMinOffset()
```

Returns the thumb offset for its minimum value.

**Returns:**

the thumb offset for its minimum value.

---

### getValue()

```
public int getValue()
```

Gets the value of the control. Note that the recovered value is subject to the control's current behavior.

**See Also:**

`SLIDER_BEHAVIOR`, `SCROLLBAR_BEHAVIOR`

---

### setBehavior(int)

```
public void setBehavior(int behavior)
```

Sets the behavior for this `HStaticRange` .

**Parameters:**

`behavior` - the behavior for this `HStaticRange` (`SLIDER_BEHAVIOR` or `SCROLLBAR_BEHAVIOR` ).

---

### setDefaultLook(HRangeLook)

```
public static void setDefaultLook(HRangeLook look)
```

Sets the default `HLook` for further `HStaticRange` Components.

**Parameters:**

`look` - The `HLook` that will be used by default when creating a new `HStaticRange` component. Note that this parameter may be null, in which case newly created components shall not draw themselves until a non-null look is set using the `setLook(HLook)` method.

---

### setLook(HLook)

```
public void setLook(HLook hlook)
```

Sets the `HLook` for this component.

**Overrides:**

`setLook(HLook)` in class `HVisible`

**Parameters:**

`hlook` - The `HLook` that is to be used for this component. Note that this parameter may be null, in which case the component will not draw itself until a look is set.

**Throws:**

`HInvalidLookException` - If the Look is not an `HRangeLook` .

---

### setOrientation(int)

```
public void setOrientation(int orient)
```

Set the orientation of the `HStaticRange` . The orientation controls how the associated `HLook` lays out the component.

**Specified By:**

`setOrientation(int)` in interface `HOrientable`

**Parameters:**

`orient` - one of `ORIENT_LEFT_TO_RIGHT` , `ORIENT_RIGHT_TO_LEFT` , `ORIENT_TOP_TO_BOTTOM` , or `ORIENT_BOTTOM_TO_TOP` .

---

**setRange(int, int)**

```
public boolean setRange(int minimum, int maximum)
```

Sets the range of values for the control.

**Parameters:**

`minimum` - The minimum value of the range control

`maximum` - The maximum value of the range control

**Returns:**

Indicates if the min and max values have been set correctly. Returns false if the minimum value is greater than or equal to the maximum value, otherwise returns true

---

**setThumbOffsets(int, int)**

```
public void setThumbOffsets(int minOffset, int maxOffset)
```

Set the offsets for the "thumb" area on this range control. The "thumb" is then drawn from (value - minOffset), to (value + maxOffset) positions outside of the HRange values [minimum:maximum] are clipped to the closest value.

There is no requirement that minOffset == maxOffset. For example, both offsets may be zero, yielding a thermometer-like range object. All measurements are in the same units as the minimum / maximum values on the `HStaticRange` object. The size of the "thumb" is the application author's responsibility. By default the "thumb" does not affect the range over which the value of the `HStaticRange` may be modified. It is recommended that the `HRangeLook` provides mechanisms to denote the value of the `HStaticRange`, in addition to indicating the extent of the thumb as defined by the offsets.

**See Also:**

`setBehavior(int)`

---

**setValue(int)**

```
public void setValue(int value)
```

Sets the value of the control, subject to its current behavior.

**Parameters:**

`value` - the value to which the control should be set.

**See Also:**

`SLIDER_BEHAVIOR`, `SCROLLBAR_BEHAVIOR`

# org.havi.ui

## HStaticText

### Syntax

public class HStaticText extends HVisible implements HNoInputPreferred

```

java.lang.Object
|
+--java.awt.Component
|
|   +--HComponent
|   |
|   |   +--HVisible
|   |   |
|   |   |   +--org.havi.ui.HStaticText

```

### Direct Known Subclasses:

HText

### All Implemented Interfaces:

HMatteLayer, HNoInputPreferred, HState, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, TestOpacity

### Description

The `HStaticText` is a user interface component used to display static read-only textual content which does *not* permit the user to navigate (focus) upon it. By default it uses the `HTextLook` to render itself. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
height	height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

text-Normal	The string to be used as the content for the <code>NORMAL_STATE</code> state of this component.	null	<code>setTextContent (String, int)</code>	<code>getTextContent (int)</code>
tlm	The text layout manager responsible for text formatting.	An <code>HDefaultTextLayoutManager</code> object.	<code>setTextLayoutManager (HTextLayoutManager)</code>	<code>getTextLayoutManager ()</code>
font	The font for this component.	---	<code>java.awt.Component#setFont</code>	<code>java.awt.Component#getFont</code>
background	The background color for this component.	---	<code>java.awt.Component#getBackground</code>	<code>java.awt.Component#setBackground</code>
foreground	The foreground color for this component.	---	<code>java.awt.Component#getForeground</code>	<code>java.awt.Component#setForeground</code>

#### Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte ( <code>HMatte</code> ).	none (i.e. <code>getMatte()</code> returns null)	<code>setMatte (HMatte )</code>	<code>getMatte ()</code>
The text layout manager responsible for text formatting.	An <code>HDefaultTextLayoutManager</code> object.	<code>setTextLayoutManager (HTextLayoutManager)</code>	<code>getTextLayoutManager ()</code>
The background painting mode	<code>NO_BACKGROUND_FILL</code>	<code>setBackgroundMode (int)</code>	<code>setBackgroundMode ()</code>
The default preferred size	not set (i.e. null) unless specified by width and height parameters	<code>setDefaultSize (Dimension)</code>	<code>getDefaultSize ()</code>
The horizontal content alignment	<code>HALIGN_CENTER</code>	<code>setHorizontalAlignment (int)</code>	<code>getHorizontalAlignment ()</code>
The vertical content alignment	<code>VALIGN_CENTER</code>	<code>setVerticalAlignment (int)</code>	<code>getVerticalAlignment ()</code>
The content scaling mode	<code>RESIZE_NONE</code>	<code>setResizeMode (int)</code>	<code>getResizeMode ()</code>
The default "look" for this class.	A platform specific <code>HTextLook</code>	<code>setDefaultLook (HTextLook)</code>	<code>getDefaultLook ()</code>

The "look" for this object.

The `HTextLook` returned from `setLook(HLook)` `getLook()`  
`HStaticText.getDefaultLook`  
when this object was created.

### See Also:

`HDefaultTextLayoutManager`, `HTextLayoutManager`

## Constructors

---

### **HStaticText()**

```
public HStaticText()
```

Creates an `HStaticText` object. See the class description for details of constructor parameters and default values.

---

### **HStaticText(String)**

```
public HStaticText(java.lang.String textNormal)
```

Creates an `HStaticText` object. See the class description for details of constructor parameters and default values.

---

### **HStaticText(String, Font, Color, Color, HTextLayoutManager)**

```
public HStaticText(java.lang.String textNormal, java.awt.Font font,  
                  java.awt.Color foreground, java.awt.Color background, HTextLayoutManager tlm)
```

Creates an `HStaticText` object. See the class description for details of constructor parameters and default values.

---

### **HStaticText(String, int, int, int, int)**

```
public HStaticText(java.lang.String textNormal, int x, int y, int width, int height)
```

Creates an `HStaticText` object. See the class description for details of constructor parameters and default values.

---

### **HStaticText(String, int, int, int, int, Font, Color, Color, HTextLayoutManager)**

```
public HStaticText(java.lang.String textNormal, int x, int y, int width, int height,  
                  java.awt.Font font, java.awt.Color foreground, java.awt.Color background,  
                  HTextLayoutManager tlm)
```

Creates an `HStaticText` object. See the class description for details of constructor parameters and default values.

## Methods



---

**getDefaultLook()**

```
public static HTextLook getDefaultLook()
```

Returns the currently set default `HLook` for `HStaticText` components.

**Returns:**

The `HLook` that is used by default when creating a new `HStaticText` component.

---

**setDefaultLook(HTextLook)**

```
public static void setDefaultLook(HTextLook hlook)
```

Sets the default `HLook` for further `HStaticText` Components.

**Parameters:**

`hlook` - The `HLook` that will be used by default when creating a new `HStaticText` component. Note that this parameter may be null, in which case newly created components shall not draw themselves until a non-null look is set using the `setLook(HLook)` method.

---

**setLook(HLook)**

```
public void setLook(HLook hlook)
```

Sets the `HLook` for this component.

**Overrides:**

`setLook(HLook)` in class `HVisible`

**Parameters:**

`hlook` - The `HLook` that is to be used for this component. Note that this parameter may be null, in which case the component will not draw itself until a look is set.

**Throws:**

`HInvalidLookException` - if the Look is not an `HTextLook`.

org.havi.ui

# HStillImageBackgroundConfiguration

## Syntax

```
public class HStillImageBackgroundConfiguration extends HBackgroundConfiguration
```

```
java.lang.Object
|
+--HScreenConfiguration
|
+--HBackgroundConfiguration
|
+--org.havi.ui.HStillImageBackgroundConfiguration
```

## Description

This class represents a background configuration which supports the installation of still images. The platform using the HAVi user-interface specification must specify which image formats are supported. The `java.awt.Image` class is intentionally not used in order to allow the support of image formats which carry sufficient restrictions that expressing them through the API of that class would require extensive use of runtime errors. One specific example of this is MPEG I-frames. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Constructors

### HStillImageBackgroundConfiguration()

```
protected HStillImageBackgroundConfiguration()
```

It is not intended that applications should directly construct `HStillImageBackgroundConfiguration` objects.

Creates an `HStillImageBackgroundConfiguration` object. See the class description for details of constructor parameters and default values.

## Methods

---

### **displayImage(HBackgroundImage)**

```
public void displayImage(HBackgroundImage image)
```

Display an image. If the data for the image has not been loaded then this method will block while the data is loaded. It is platform dependent whether this image is scaled to fit or whether it is cropped (where too large) or repeated (where too small). The position of the image is platform-dependent. If the platform does not scale the image to fit, the previous color set using `setColor(Color)` is shown in the areas where no image is displayed. If no color has been set what is shown in this area is platform dependent. What is displayed while the image is loading is implementation specific.

Note that the image may be removed by calling the `setColor(Color)` method.

If the `image` parameter is null a `java.lang.NullPointerException` is thrown.

**Parameters:**

`image` - the image to display.

**Throws:**

`java.io.IOException` - if the data for the `HBackgroundImage` is not loaded and loading the data is impossible or fails.

`java.lang.IllegalArgumentException` - if the `HBackgroundImage` does not contain an image in a supported image encoding format

`HPermissionDeniedException` - if the `HBackgroundDevice` concerned is not reserved.

`HConfigurationException` - if the `HStillImageBackgroundConfiguration` is not the currently set configuration for its `HBackgroundDevice`.

---

### **displayImage(HBackgroundImage, HScreenRectangle)**

```
public void displayImage(HBackgroundImage image, HScreenRectangle r)
```

Display an image to cover a particular area of the screen. If the data for the image has not been loaded then this method will block while the data is loaded. It is platform dependent whether this image is scaled to fit or whether it is cropped (where too large) or repeated (where too small). The position of the image within the rectangle is platform-dependent. If the platform does not scale the image to fit, or the rectangle does not cover the entire display area, the previous color set using `setColor(Color)` is shown in the areas where no image is displayed. If no color has been set what is shown in this area is platform dependent.

Note that the image may be removed by calling the `setColor(Color)` method.

If either or both parameters are null a `java.lang.NullPointerException` is thrown.

**Parameters:**

`image` - the image to display

`r` - the area of the screen to cover with the image

**Throws:**

`java.io.IOException` - if the data for the `HBackgroundImage` is not loaded and loading the data is impossible or fails.

`java.lang.IllegalArgumentException` - if the `HBackgroundImage` does not contain an image in a supported image encoding format

`HPermissionDeniedException` - if the `HBackgroundDevice` concerned is not reserved.

`HConfigurationException` - if the `HStillImageBackgroundConfiguration` is not the currently set configuration for its `HBackgroundDevice`.

---

**setColor(Color)**

```
public void setColor(java.awt.Color color)
```

Set the current color of this background. On platforms where there is a sub-class of `java.awt.Color` supporting transparency of any kind, passing an object representing a non-opaque color is illegal. Platforms with a limited color resolution for backgrounds may approximate this value to the nearest available. The `getColor()` method will return the actual value used.

Note that calling this method will clear any image currently displayed by the `HBackgroundDevice`.

**Overrides:**

```
setColor(Color) in class HBackgroundConfiguration
```

**Parameters:**

`color` - the color to be used for the background

**Throws:**

`HPermissionDeniedException` - if this `HBackgroundDevice` does not have the right to control the background

`HConfigurationException` - if the color specified is illegal for this platform.

# org.havi.ui

# HSwitchable

## Syntax

```
public interface HSwitchable extends HActionable
```

## All Superinterfaces:

```
HActionable, HActionInputPreferred, HNavigable, HNavigationInputPreferred
```

## All Known Implementing Classes:

```
HToggleButton
```

## Description

This interface is implemented for all user interface components that can be actioned such that they "toggle" on and off and maintain the chosen state.

### Event Behavior

Subclasses of `HComponent` which implement `HSwitchable` must respond to `HFocusEvent` and `HActionEvent` events.

Applications should assume that classes which implement `HSwitchable` can generate events of the types `HFocusEvent` and `HActionEvent` in response to other types of input event.

An application may add one or more `HActionListener` listeners to the component. The `actionPerformed` method of the `HActionListener` is invoked whenever the `HSwitchable` is actioned.

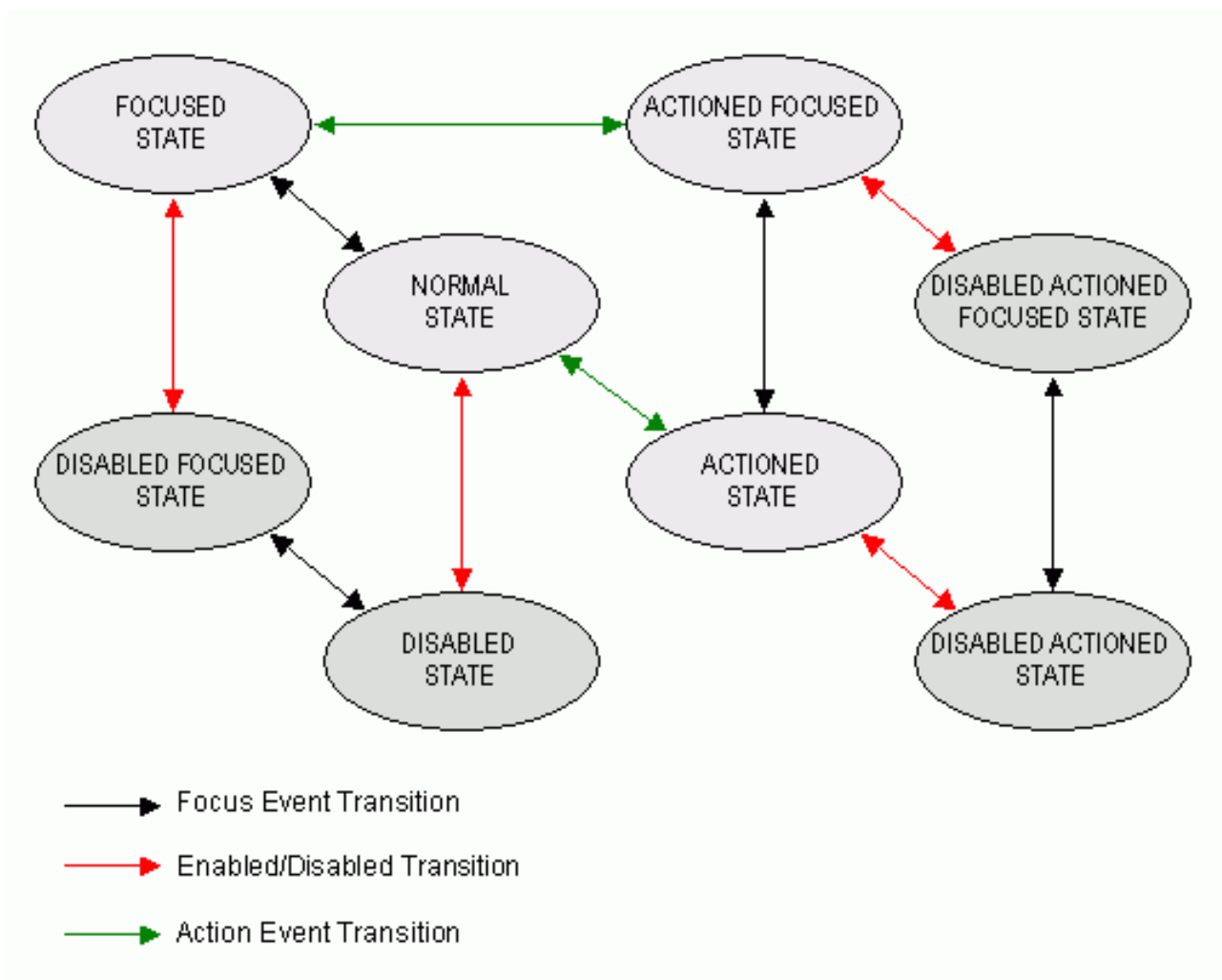
HAVi action events are discussed in detail in the `HActionInputPreferred` interface description.

### Interaction States

The following interaction states are valid for this `HSwitchable` component:

- `NORMAL_STATE`
- `FOCUSED_STATE`
- `ACTIONED_STATE`
- `ACTIONED_FOCUSED_STATE`
- `DISABLED_STATE`
- `DISABLED_FOCUSED_STATE`
- `DISABLED_ACTIONED_STATE`
- `DISABLED_ACTIONED_FOCUSED_STATE`

The state machine diagram below shows the valid state transitions for an `HSwitchable` component.



Unlike `HActionable` components there are no automatic transitions to other states. Actioned states (i.e. those with the `ACTIONED_STATE_BIT` bit set may persist after any registered `HActionListener` listeners have been called, until a further `HActionEvent` is received.

#### Platform Classes

The following HAVi platform classes implement or inherit the `HSwitchable` interface. These classes shall all generate both `HFocusEvent` and `HActionEvent` events in addition to any other events specified in the respective class descriptions.

- `HToggleButton`

#### See Also:

`HNavigable`, `HActionable`, `HActionInputPreferred`, `HActionEvent`, `HActionListener`

## Methods

---

**getSwitchableState()**

```
public boolean getSwitchableState()
```

Returns the current switchable state of this `HSwitchable` .

**Returns:**

the current switchable state of this `HSwitchable` .

---

**getUnsetActionSound()**

```
public HSound getUnsetActionSound()
```

Get the sound to be played when the interaction state of the `HSwitchable` makes the following transitions:

- `ACTIONED_STATE` to `NORMAL_STATE`
- `ACTIONED_FOCUSED_STATE` to `FOCUSED_STATE`

**Returns:**

the sound to be played when the `HSwitchable` transitions from an actioned state.

---

**setSwitchableState(boolean)**

```
public void setSwitchableState(boolean state)
```

Sets the current state of the button. Note that `ActionListeners` are only called when an `ACTION_PERFORMED` event is received, or if they are called directly, e.g. via `processActionEvent`, they are not called by `setSwitchableState(boolean)` .

---

**setUnsetActionSound(HSound)**

```
public void setUnsetActionSound(HSound sound)
```

Associate a sound to be played when the interaction state of the `HSwitchable` makes the following transitions:

- `ACTIONED_STATE` to `NORMAL_STATE`
- `ACTIONED_FOCUSED_STATE` to `FOCUSED_STATE`

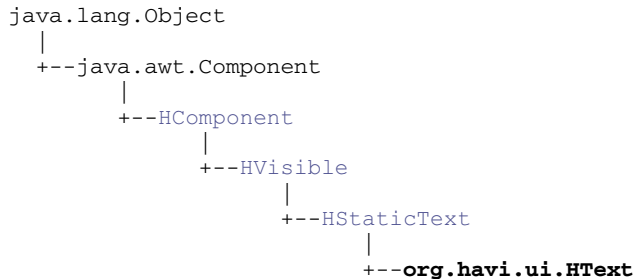
**Parameters:**

`sound` - a sound to be played when the `HSwitchable` transitions from an actioned state. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound` .

# org.havi.ui HText

## Syntax

public class HText extends HStaticText implements HNavigable



## Direct Known Subclasses:

HTextButton

## All Implemented Interfaces:

HMatteLayer, HNavigable, HNavigationInputPreferred, HNoInputPreferred, HState, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, TestOpacity

## Description

The `HText` is a user interface component used to display static read-only textual content (as `HStaticText`) which additionally enables a user to navigate to it, i.e. it can have the input focus. By default it uses the `HTextLook` to render itself. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
height	height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds



text	The text to be used as the content for every permissible state of this component.	null	setTextContent (String, int)	getTextContent (int)
text-Normal	The string to be used as the content for the <code>NORMAL_STATE</code> state of this component.	null	setTextContent (String, int)	getTextContent (int)
text-Focus	The string to be used as the content for the focused state of this component.	null	setTextContent (String, int)	getTextContent (int)
tlm	The text layout manager responsible for text formatting.	An <code>HDefaultTextLayoutManager</code> object.	setTextLayoutManager (HTextLayoutManager)	getTextLayoutManager ()
font	The font for this component.	---	java.awt.Component#setFont	java.awt.Component#getFont
background	The background color for this component.	---	java.awt.Component#getBackground	java.awt.Component#setBackground
foreground	The foreground color for this component.	---	java.awt.Component#getForeground	java.awt.Component#setForeground

#### Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte ( <code>HMatte</code> ).	none (i.e. <code>getMatte()</code> returns null)	<code>setMatte (HMatte)</code>	<code>getMatte ()</code>
The text layout manager responsible for text formatting.	An <code>HDefaultTextLayoutManager</code> object.	<code>setTextLayoutManager (HTextLayoutManager)</code>	<code>getTextLayoutManager ()</code>
The background painting mode	<code>NO_BACKGROUND_FILL</code>	<code>setBackgroundMode (int)</code>	<code>getBackgroundMode ()</code>
The default preferred size	not set (i.e. null) unless specified by width and height parameters	<code>setDefaultSize (Dimension)</code>	<code>getDefaultSize ()</code>
The horizontal content alignment	<code>HALIGN_CENTER</code>	<code>setHorizontalAlignment (int)</code>	<code>getHorizontalAlignment ()</code>
The vertical content alignment	<code>VALIGN_CENTER</code>	<code>setVerticalAlignment (int)</code>	<code>getVerticalAlignment ()</code>

The content scaling mode	<code>RESIZE_NONE</code>	<code>setResizeMode(int)</code>	<code>getResizeMode()</code>
The default "look" for this class.	A platform specific <code>HTextLook</code>	<code>setDefaultLook(HTextLook)</code>	<code>getDefaultLook()</code>
The "look" for this object.	The <code>HTextLook</code> returned from <code>HText.getDefaultLook</code> when this object was created.	<code>setLook(HLook)</code>	<code>getLook()</code>
The gain focus sound.	<code>null</code>	<code>setGainFocusSound(HSound)</code>	<code>getGainFocusSound()</code>
The lose focus sound.	<code>null</code>	<code>setLoseFocusSound(HSound)</code>	<code>getLoseFocusSound()</code>

**See Also:**

`HStaticText`, `HDefaultTextLayoutManager`, `HTextLayoutManager`, `HNavigable`

## Constructors

---

**HText()**

```
public HText()
```

Creates an `HText` object. See the class description for details of constructor parameters and default values.

---

**HText(String)**

```
public HText(java.lang.String text)
```

Creates an `HText` object. See the class description for details of constructor parameters and default values.

---

**HText(String, Font, Color, Color, HTextLayoutManager)**

```
public HText(java.lang.String text, java.awt.Font font, java.awt.Color foreground,
             java.awt.Color background, HTextLayoutManager tlm)
```

Creates an `HText` object. See the class description for details of constructor parameters and default values.

---

**HText(String, int, int, int, int)**

```
public HText(java.lang.String text, int x, int y, int width, int height)
```

Creates an `HText` object. See the class description for details of constructor parameters and default values.

---

**HText(String, int, int, int, int, Font, Color, Color, HTextLayoutManager)**

```
public HText(java.lang.String text, int x, int y, int width, int height,  
             java.awt.Font font, java.awt.Color foreground, java.awt.Color background,  
             HTextLayoutManager tlm)
```

Creates an `HText` object. See the class description for details of constructor parameters and default values.

---

**HText(String, String)**

```
public HText(java.lang.String textNormal, java.lang.String textFocus)
```

Creates an `HText` object. See the class description for details of constructor parameters and default values.

---

**HText(String, String, Font, Color, Color, HTextLayoutManager)**

```
public HText(java.lang.String textNormal, java.lang.String textFocus, java.awt.Font font,  
             java.awt.Color foreground, java.awt.Color background, HTextLayoutManager tlm)
```

Creates an `HText` object. See the class description for details of constructor parameters and default values.

---

**HText(String, String, int, int, int, int)**

```
public HText(java.lang.String textNormal, java.lang.String textFocus, int x, int y,  
             int width, int height)
```

Creates an `HText` object. See the class description for details of constructor parameters and default values.

---

**HText(String, String, int, int, int, int, Font, Color, Color, HTextLayoutManager)**

```
public HText(java.lang.String textNormal, java.lang.String textFocus, int x, int y,  
             int width, int height, java.awt.Font font, java.awt.Color foreground,  
             java.awt.Color background, HTextLayoutManager tlm)
```

Creates an `HText` object. See the class description for details of constructor parameters and default values.

## Methods

---

**getDefaultLook()**

```
public static HTextLook getDefaultLook()
```

Returns the currently set default `HLook` for `HText` components.

**Returns:**

The `HLook` that is used by default when creating a new `HText` component.

---

**getGainFocusSound()**

```
public HSound getGainFocusSound()
```

Get the sound associated with the gain focus event.

**Specified By:**

`getGainFocusSound()` in interface `HNavigable`

**Returns:**

The sound played when the component gains focus. If no sound is associated with gaining focus, then null shall be returned.

---

**getLoseFocusSound()**

```
public HSound getLoseFocusSound()
```

Get the sound associated with the lost focus event.

**Specified By:**

`getLoseFocusSound()` in interface `HNavigable`

**Returns:**

The sound played when the component loses focus. If no sound is associated with losing focus, then null shall be returned.

---

**getMove(int)**

```
public HNavigable getMove(int keyCode)
```

Provides the `HNavigable` object that is navigated to when a particular key is pressed.

**Specified By:**

`getMove(int)` in interface `HNavigable`

**Parameters:**

`keyCode` - The key code of the pressed key.

**Returns:**

Returns the `HNavigable` object, or if no `HNavigable` is associated with the `keyCode` then returns null.

---

**getNavigationKeys()**

```
public int[] getNavigationKeys()
```

Retrieve the set of key codes which this component maps to navigation targets.

**Specified By:**

`getNavigationKeys()` in interface `HNavigationInputPreferred`

**Returns:**

an array of key codes, or null if no navigation targets are set on this component.

---

**isSelected()**

```
public boolean isSelected()
```

Indicates if this component has focus.

**Specified By:**

`isSelected()` in interface `HNavigable`

**Returns:**

true if the component has focus, otherwise returns false.

---

**processHFocusEvent(HFocusEvent)**

```
public void processHFocusEvent(HFocusEvent evt)
```

Process an `HFocusEvent` sent to this `HText`.

**Specified By:**

```
processHFocusEvent(HFocusEvent) in interface HNavigationInputPreferred
```

**Parameters:**

`evt` - the `HFocusEvent` to process.

---

**setDefaultLook(HTextLook)**

```
public static void setDefaultLook(HTextLook look)
```

Sets the default `HLook` for further `HText` Components.

**Parameters:**

`look` - The `HLook` that will be used by default when creating a new `HText` component. Note that this parameter may be null, in which case newly created components shall not draw themselves until a non-null look is set using the `setLook(HLook)` method.

---

**setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)**

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
                             HNavigable right)
```

Set the focus control for an `HNavigable` component. Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is a convenience function for application programmers where a standard up, down, left and right focus traversal between components is required.

Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

Note that this API does not prevent the creation of "isolated" `HNavigable` components --- authors should endeavor to avoid confusing the user.

**Specified By:**

```
setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) in
interface HNavigable
```

**Parameters:**

`up` - The `HNavigable` component to move to, when the user generates a `VK_UP` KeyEvent. If there is no `HNavigable` component to move "up" to, then null should be specified.

`down` - The `HNavigable` component to move to, when the user generates a `VK_DOWN` KeyEvent. If there is no `HNavigable` component to move "down" to, then null should be specified.

`left` - The `HNavigable` component to move to, when the user generates a `VK_LEFT` KeyEvent. If there is no `HNavigable` component to move "left" to, then null should be specified.

`right` - The `HNavigable` component to move to, when the user generates a `VK_RIGHT` KeyEvent. If there is no `HNavigable` component to move "right" to, then null should be specified.

---

**setGainFocusSound(HSound)**

```
public void setGainFocusSound(HSound sound)
```

Associate a sound with gaining focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_GAINED`. This sound will start to be played when an object implementing this interface gains focus. It is not guaranteed to be played to completion. If the object implementing this interface loses focus before the audio completes playing, the audio will be truncated. Applications wishing to ensure the audio is always played to completion must implement special logic to slow down the focus transitions.

By default, an `HNavigable` object does not have any gain focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

**Specified By:**

`setGainFocusSound(HSound)` in interface `HNavigable`

**Parameters:**

`sound` - the sound to be played, when the component gains focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

### **setLoseFocusSound(HSound)**

```
public void setLoseFocusSound(HSound sound)
```

Associate a sound with losing focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_LOST`. This sound will start to be played when an object implementing this interface loses focus. It is not guaranteed to be played to completion. It is implementation dependent whether and when this sound will be truncated by any gain focus sound played by the next object to gain focus.

By default, an `HNavigable` object does not have any lose focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

**Specified By:**

`setLoseFocusSound(HSound)` in interface `HNavigable`

**Parameters:**

`sound` - the sound to be played, when the component loses focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

### **setMove(int, HNavigable)**

```
public void setMove(int keyCode, HNavigable target)
```

Defines the navigation path from the current `HNavigable` to another `HNavigable` when a particular key is pressed.

Note that `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

**Specified By:**

`setMove(int, HNavigable)` in interface `HNavigable`

**Parameters:**

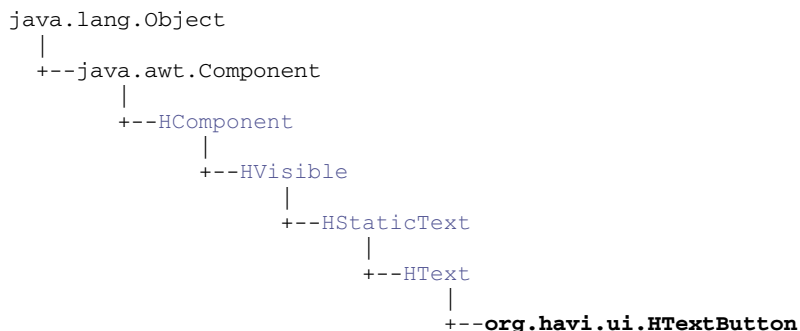
`keyCode` - The key code of the pressed key. Any numerical keycode is allowed, but the platform may not be able to generate all keycodes. Application authors should only use keys for which `HRCapabilities.isSupported()` or `HKeyCapabilities.isSupported()` returns true.

`target` - The target `HNavigable` object that should be navigated to. If a target is to be removed from a particular navigation path, then `null` should be specified.

# org.havi.ui HTextButton

## Syntax

```
public class HTextButton extends HText implements HActionable
```



## All Implemented Interfaces:

`HActionable`, `HActionInputPreferred`, `HMatteLayer`, `HNavigable`,  
`HNavigationInputPreferred`, `HNoInputPreferred`, `HState`,  
`java.awt.image.ImageObserver`, `java.awt.MenuContainer`, `java.io.Serializable`, `TestOpacity`

## Description

The `HTextButton` is a user interface component representing a push-button with static read-only textual content. This component can be navigated to, i.e. it can have the input focus, and it can also be actioned as defined by the `HActionable` interface. By default it uses the `HTextLook` class to render itself. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
width	width of this component in pixels (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>

height	height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
text	The text to be used as the content for every permissible state of this component.	null	setTextContent (String, int)	getTextContent (int)
text-Normal	The string to be used as the content for the <code>NORMAL_STATE</code> state of this component.	null	setTextContent (String, int)	getTextContent (int)
text-Focus	The string to be used as the content for the focused state of this component.	null	setTextContent (String, int)	getTextContent (int)
text-Action	The string to be used as the content for the actioned states of this component.	null	setTextContent (String, int)	getTextContent (int)
tlm	The text layout manager responsible for text formatting.	An <code>HDefaultTextLayoutManager</code> object.	setTextLayoutManager (HTextLayoutManager)	getTextLayoutManager ()
font	The font for this component.	---	java.awt.Component#setFont	java.awt.Component#getFont
background	The background color for this component.	---	java.awt.Component#getBackground	java.awt.Component#setBackground
foreground	The foreground color for this component.	---	java.awt.Component#getForeground	java.awt.Component#setForeground

#### Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte ( <code>HMatte</code> ).	none (i.e. <code>getMatte()</code> returns null)	<code>setMatte (HMatte )</code>	<code>getMatte ()</code>
The text layout manager responsible for text formatting.	An <code>HDefaultTextLayoutManager</code> object.	<code>setTextLayoutManager (HTextLayoutManager)</code>	<code>getTextLayoutManager ()</code>
The background painting mode	<code>NO_BACKGROUND_FILL</code>	<code>setBackgroundMode (int)</code>	<code>setBackgroundMode ()</code>
The default preferred size	not set (i.e. null) unless specified by width and height parameters	<code>setDefaultSize ( Dimension)</code>	<code>getDefaultSize ()</code>



The horizontal content alignment	<code>HALIGN_CENTER</code>	<code>setHorizontalAl ignment(int)</code>	<code>getHorizontalAl ignment()</code>
The vertical content alignment	<code>VALIGN_CENTER</code>	<code>setVerticalAlig nment(int)</code>	<code>getVerticalAlig nment()</code>
The content scaling mode	<code>RESIZE_NONE</code>	<code>setResizeMode(i nt)</code>	<code>getResizeMode()</code>
The default "look" for this class.	A platform specific <code>HTextLook</code>	<code>setDefaultLook( HTextLook)</code>	<code>getDefaultLook( )</code>
The "look" for this object.	The <code>HTextLook</code> returned from <code>HTextButton.getDefaultLook</code> when this object was created.	<code>setLook(HLook)</code>	<code>getLook()</code>
The gain focus sound.	<code>null</code>	<code>setGainFocusSou nd(HSound)</code>	<code>getGainFocusSou nd()</code>
The lose focus sound.	<code>null</code>	<code>setLoseFocusSou nd(HSound)</code>	<code>getLoseFocusSou nd()</code>
The action sound.	<code>null</code>	<code>setActionSound( HSound)</code>	<code>getActionSound( )</code>

**See Also:**

`HStaticText`, `HText`, `HDefaultTextLayoutManager`, `HTextLayoutManager`, `HNavigable`, `HActionable`

## Constructors

---

**HTextButton()**

```
public HTextButton()
```

Creates an `HTextButton` object. See the class description for details of constructor parameters and default values.

---

**HTextButton(String)**

```
public HTextButton(java.lang.String text)
```

Creates an `HTextButton` object. See the class description for details of constructor parameters and default values.

---

**HTextButton(String, Font, Color, Color, HTextLayoutManager)**

```
public HTextButton(java.lang.String text, java.awt.Font font, java.awt.Color foreground,  
java.awt.Color background, HTextLayoutManager tlm)
```

Creates an `HTextButton` object. See the class description for details of constructor parameters and default values.

---

**HTextButton(String, int, int, int, int)**

```
public HTextButton(java.lang.String text, int x, int y, int width, int height)
```

Creates an `HTextButton` object. See the class description for details of constructor parameters and default values.

---

**HTextButton(String, int, int, int, int, Font, Color, Color, HTextLayoutManager)**

```
public HTextButton(java.lang.String text, int x, int y, int width, int height,
    java.awt.Font font, java.awt.Color foreground, java.awt.Color background,
    HTextLayoutManager tlm)
```

Creates an `HTextButton` object. See the class description for details of constructor parameters and default values.

## Methods

---

**addActionListener(HActionListener)**

```
public void addActionListener(HActionListener l)
```

Adds the specified `HActionListener` to receive `HActionEvent` events sent from this `HTextButton`. If the listener has already been added further calls will add further references to the listener, which will then receive multiple copies of a single event.

**Specified By:**

`addActionListener(HActionListener)` in interface `HActionable`

**Parameters:**

l - the `HActionListener`.

---

**getActionCommand()**

```
public java.lang.String getActionCommand()
```

Gets the command name for the `HActionEvent` event fired by this `HTextButton`.

**Specified By:**

`getActionCommand()` in interface `HActionable`

**Returns:**

A `String` representing the command name of the `HActionEvent` fired by this `HTextButton`.

**See Also:**

`getActionCommand()`

---

**getActionSound()**

```
public HSound getActionSound()
```

Associate a sound to be played when the interaction state of the `HTextButton` makes the following transitions:

- `_NORMAL_STATE` to `_ACTIONED_STATE`
- `_FOCUSED_STATE` to `_ACTIONED_FOCUSED_STATE`

**Specified By:**

`getActionSound()` in interface `HActionable`

**Returns:**

The sound played when the component is actioned.

---

**getDefaultLook()**

```
public static HTextLook getDefaultLook()
```

Returns the currently set default `HLook` for `HTextButton` components.

**Returns:**

The `HLook` that is used by default when creating a new `HTextButton`.

---

**getGainFocusSound()**

```
public HSound getGainFocusSound()
```

Get the sound associated with the gain focus event.

**Specified By:**

`getGainFocusSound()` in interface `HNavigable`

**Overrides:**

`getGainFocusSound()` in class `HText`

**Returns:**

The sound played when the component gains focus. If no sound is associated with gaining focus, then null shall be returned.

---

**getLoseFocusSound()**

```
public HSound getLoseFocusSound()
```

Get the sound associated with the lost focus event.

**Specified By:**

`getLoseFocusSound()` in interface `HNavigable`

**Overrides:**

`getLoseFocusSound()` in class `HText`

**Returns:**

The sound played when the component loses focus. If no sound is associated with losing focus, then null shall be returned.

---

**getMove(int)**

```
public HNavigable getMove(int keyCode)
```

Provides the `HNavigable` object that is navigated to when a particular key is pressed.

**Specified By:**

`getMove(int)` in interface `HNavigable`

**Overrides:**

`getMove(int)` in class `HText`

**Parameters:**

`keyCode` - The key code of the pressed key.

---

**Returns:**

Returns the `HNavigable` object, or if no `HNavigable` is associated with the `keyCode` then returns `null`.

---

**getNavigationKeys()**

```
public int[] getNavigationKeys()
```

Retrieve the set of key codes which this component maps to navigation targets.

**Specified By:**

```
getNavigationKeys() in interface HNavigationInputPreferred
```

**Overrides:**

```
getNavigationKeys() in class HText
```

**Returns:**

an array of key codes, or `null` if no navigation targets are set on this component.

---

**isSelected()**

```
public boolean isSelected()
```

Indicates if this component has focus.

**Specified By:**

```
isSelected() in interface HNavigable
```

**Overrides:**

```
isSelected() in class HText
```

**Returns:**

`true` if the component has focus, otherwise returns `false`.

---

**processHActionEvent(HActionEvent)**

```
public void processHActionEvent(HActionEvent evt)
```

Process an `HActionEvent` sent to this `HTextButton`.

**Specified By:**

```
processHActionEvent(HActionEvent) in interface HActionInputPreferred
```

**Parameters:**

`evt` - the `HActionEvent` to process.

---

**processHFocusEvent(HFocusEvent)**

```
public void processHFocusEvent(HFocusEvent evt)
```

Process an `HFocusEvent` sent to this `HTextButton`.

**Specified By:**

```
processHFocusEvent(HFocusEvent) in interface HNavigationInputPreferred
```

**Overrides:**

```
processHFocusEvent(HFocusEvent) in class HText
```

**Parameters:**

`evt` - the `HFocusEvent` to process.

---

**removeHActionListener(HActionListener)**

```
public void removeHActionListener(HActionListener l)
```

Removes the specified `HActionListener` so that it no longer receives `HActionEvent` events from this `HTextButton`. If the specified listener is not registered, the method has no effect. If multiple references to a single listener have been registered it should be noted that this method will only remove one reference per call.

**Specified By:**

`removeHActionListener(HActionListener)` in interface `HActionable`

**Parameters:**

l - the `HActionListener`.

---

**setActionCommand(String)**

```
public void setActionCommand(java.lang.String command)
```

Sets the command name for the `HActionEvent` event fired by this `HTextButton`.

**Specified By:**

`setActionCommand(String)` in interface `HActionable`

**Parameters:**

command - a `String` used to set the action command.

**See Also:**

`getActionCommand()`

---

**setActionSound(HSound)**

```
public void setActionSound(HSound sound)
```

Associate a sound to be played when the interaction state of the `HTextButton` makes the following transitions:

- `_NORMAL_STATE` to `_ACTIONED_STATE`
- `_FOCUSED_STATE` to `_ACTIONED_FOCUSED_STATE`

**Specified By:**

`setActionSound(HSound)` in interface `HActionable`

**Parameters:**

sound - the sound to be played, when the component is actioned. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

---

**setDefaultLook(HTextLook)**

```
public static void setDefaultLook(HTextLook hlook)
```

Sets the default `HLook` for further `HTextButton` components.

**Parameters:**

hlook - The `HLook` that will be used by default when creating a new `HTextButton`. Note that this parameter may be null, in which case newly created components shall not draw themselves until a non-null look is set using the `setLook(HLook)` method.

---

**setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)**

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
    HNavigable right)
```

Set the focus control for an `HNavigable` component. Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is a convenience function for application programmers where a standard up, down, left and right focus traversal between components is required.

Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

Note that this API does not prevent the creation of "isolated" `HNavigable` components --- authors should endeavor to avoid confusing the user.

**Specified By:**

`setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` in interface `HNavigable`

**Overrides:**

`setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` in class `HText`

**Parameters:**

`up` - The `HNavigable` component to move to, when the user generates a `VK_UP` KeyEvent. If there is no `HNavigable` component to move "up" to, then null should be specified.

`down` - The `HNavigable` component to move to, when the user generates a `VK_DOWN` KeyEvent. If there is no `HNavigable` component to move "down" to, then null should be specified.

`left` - The `HNavigable` component to move to, when the user generates a `VK_LEFT` KeyEvent. If there is no `HNavigable` component to move "left" to, then null should be specified.

`right` - The `HNavigable` component to move to, when the user generates a `VK_RIGHT` KeyEvent. If there is no `HNavigable` component to move "right" to, then null should be specified.

### **setGainFocusSound(HSound)**

```
public void setGainFocusSound(HSound sound)
```

Associate a sound with gaining focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_GAINED`. This sound will start to be played when an object implementing this interface gains focus. It is not guaranteed to be played to completion. If the object implementing this interface loses focus before the audio completes playing, the audio will be truncated. Applications wishing to ensure the audio is always played to completion must implement special logic to slow down the focus transitions.

By default, an `HNavigable` object does not have any gain focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

**Specified By:**

`setGainFocusSound(HSound)` in interface `HNavigable`

**Overrides:**

`setGainFocusSound(HSound)` in class `HText`

**Parameters:**

`sound` - the sound to be played, when the component gains focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

### **setLoseFocusSound(HSound)**

```
public void setLoseFocusSound(HSound sound)
```

Associate a sound with losing focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_LOST`. This sound will start to be played when an object implementing this interface loses focus. It is not guaranteed to be played to completion. It is implementation dependent whether and when this sound will be truncated by any gain focus sound played by the next object to gain focus.

By default, an `HNavigable` object does not have any lose focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

**Specified By:**

`setLoseFocusSound(HSound)` in interface `HNavigable`

**Overrides:**

`setLoseFocusSound(HSound)` in class `HText`

**Parameters:**

`sound` - the sound to be played, when the component loses focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

---

**setMove(int, HNavigable)**

```
public void setMove(int keyCode, HNavigable target)
```

Defines the navigation path from the current `HNavigable` to another `HNavigable` when a particular key is pressed.

Note that `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

**Specified By:**

`setMove(int, HNavigable)` in interface `HNavigable`

**Overrides:**

`setMove(int, HNavigable)` in class `HText`

**Parameters:**

`keyCode` - The key code of the pressed key. Any numerical keycode is allowed, but the platform may not be able to generate all keycodes. Application authors should only use keys for which `HRcCapabilities.isSupported()` or `HKeyCapabilities.isSupported()` returns true.

`target` - The target `HNavigable` object that should be navigated to. If a target is to be removed from a particular navigation path, then `null` should be specified.

# org.havi.ui HTextLayoutManager

## Syntax

```
public interface HTextLayoutManager
```

## All Known Implementing Classes:

```
DVBTextLayoutManager, HDefaultTextLayoutManager
```

## Description

The `HTextLayoutManager` class manages the layout and rendering on-screen of a "marked-up" string.

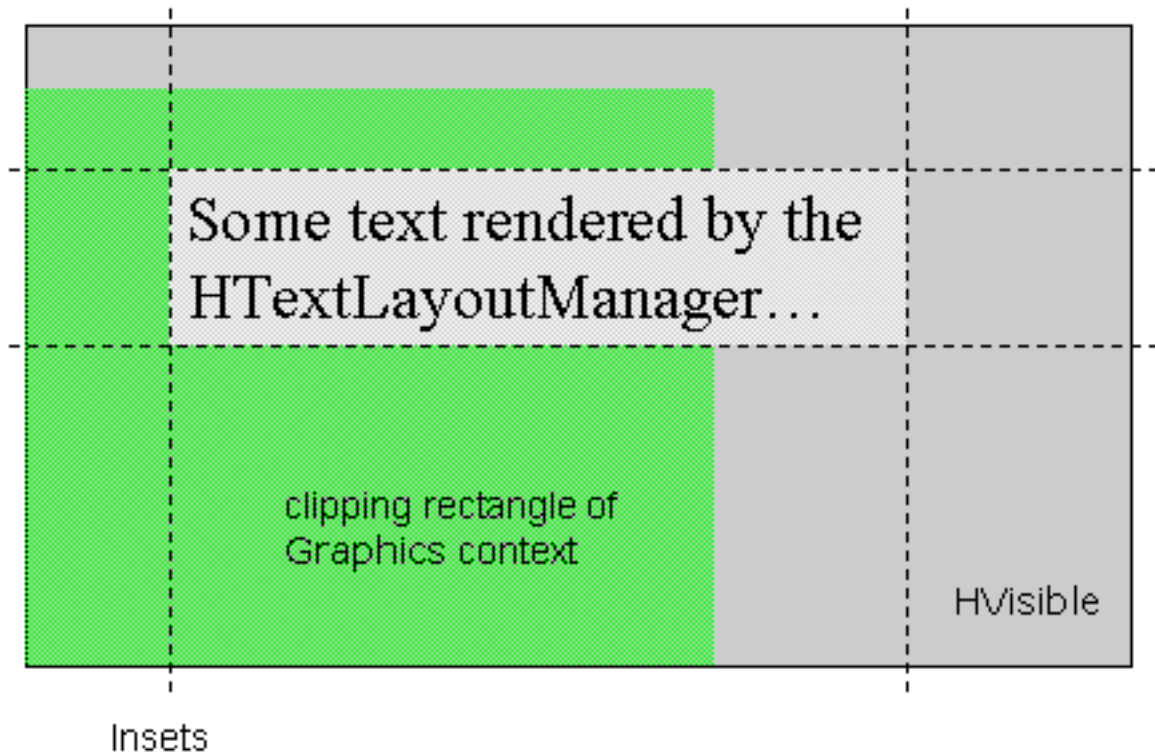
Possible implementations of `HTextLayoutManager` could enable the following behaviors:

- Interpreting basic markup, such as changing color or font, and forced line breaks.
- Providing text alignment, as in the `HDefaultTextLayoutManager`.
- Providing text wrapping policies, such as word-wrap.
- Providing text orientations, such as right-to-left, or top-to-bottom rendering.
- Providing specialized support for missing characters, or fonts.
- Providing specific language support.
- Additional text styles, such as drop capitals or "shadow" characters.

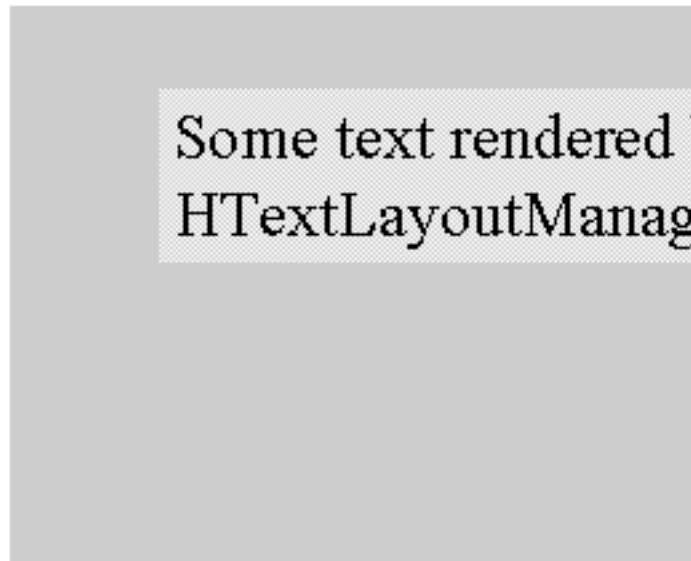
`HTextLayoutManager` supports passing a `java.awt.Insets` object as argument to the `render(String, Graphics, HVisible, Insets)` method to restrict the area in which text may be rendered. If the insets are zero, the text is rendered into the area defined by the bounds of the `HVisible` passed to the `render(String, Graphics, HVisible, Insets)` method.

The clipping rectangle of the `Graphics` object passed to the `render(String, Graphics, HVisible, Insets)` method is *not* used to determine the area in which text is rendered, as its size and position is not guaranteed to cover the entire `HVisible` - for example, when partial repainting of an `HVisible` is performed. The diagram below shows a possible scenario:





The gray area shows the bounds of the `HVisible`. The green area shows the clipping rectangle of the `Graphics` context, and the dashed lines show the insets passed to the `render(String, Graphics, HVisible, Insets)` method. The text is laid out into the rectangle defined by the `HVisible` bounds after subtracting the insets. However, only the part of the text covered by the clipping rectangle is actually drawn to the screen, as shown in the diagram below:



The behavior of the render method when the text to be rendered does not fit in the current area specified is implementation-specific.

#### See Also:

`HDefaultTextLayoutManager`

## Methods

---

### **render(String, Graphics, HVisible, Insets)**

```
public void render(java.lang.String markedUpString, java.awt.Graphics g, HVisible v,  
                  java.awt.Insets insets)
```

Render the string. The `HTextLayoutManager` should use the passed `HVisible` object to determine any additional information required to render the string, e.g. `Font`, `Color` etc.

The text should be laid out in the layout area, which is defined by the bounds of the specified `HVisible`, after subtracting the insets. If the insets are `null` the full bounding rectangle is used as the area to render text into.

The `HTextLayoutManager` should not modify the clipping rectangle of the `Graphics` object.

#### Parameters:

`markedUpString` - the string to render.

`g` - the graphics context, including a clipping rectangle which encapsulates the area within which rendering is permitted. If a valid insets value is passed to this method then text must only be rendered into the bounds of the widget after the insets are subtracted. If the insets value is `null` then text is rendered into the entire bounding area of the `HVisible`. It is implementation specific whether or not the renderer takes into account the intersection of the clipping rectangle in each case for optimization purposes.

`v` - the `HVisible` into which to render.

`insets` - the insets to determine the area in which to layout the text, or `null`.

# org.havi.ui

# HTextLook

## Syntax

```
public class HTextLook implements HLook
```

```
java.lang.Object
|
+--org.havi.ui.HTextLook
```

## All Implemented Interfaces:

```
java.lang.Cloneable, HLook
```

## Description

The `HTextLook` class is used by any `HVisible` component to display static text content. This look will be provided by the platform and the exact way in which it is rendered will be platform dependant.

The `HTextLook` class draws the content set on an `HVisible`. It uses the `getTextContent(int)` [getTextContent\(int\)](#)

In addition to the basic class behavior defined by the `HLook` interface the `HTextLook` associated with a given `HVisible` should use:

- `HVisible.getTextLayoutManager()` to determine the `HTextLayoutManager` to perform the rendering. If the returned object is null, then the text should not be rendered.

Implementations of `HTextLook` need not use the alignment mode methods on `HVisible`, since all rendering of content is deferred to the associated `HTextLayoutManager` which will use these methods accordingly.

This is the default look that is used by `HListElement`, `HStaticText` and their subclasses. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## See Also:

```
HStaticText, HDefaultTextLayoutManager
```

## Constructors

---

## HTextLook()

```
public HTextLook()
```

Creates a `HTextLook` object. See the class description for details of constructor parameters and default values.

## Methods

---

### getInsets(HVisible)

```
public java.awt.Insets getInsets(HVisible visible)
```

Determines the insets of this `HLook`, which indicate the size of the border. This area is reserved for the `HLook` to use for drawing borders around the associated `HVisible`.

**Specified By:**

`getInsets(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

the insets of this `HLook`.

---

### getMaximumSize(HVisible)

```
public java.awt.Dimension getMaximumSize(HVisible hvisible)
```

Gets the maximum size of the `HVisible` component when drawn with this `HLook`.

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook`) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If the `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) then the return value is the current size of the `HVisible` (as returned by `HVisible#getSize`).
2. If the `HLook` does not support scaling of content or no scaling is requested, and content is set then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If there is no content set then a maximum size of `[ Short.MAX_VALUE, Short.MAX_VALUE ]` is returned as a `Dimension`.

**Specified By:**

`getMaximumSize(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating this `HLook` maximum size.

**See Also:**

`getMaximumSize()`

---

**getMinimumSize(HVisible)**

```
public java.awt.Dimension getMinimumSize(HVisible hvisible)
```

Gets the minimum size of the `HVisible` component when drawn with this `HLook`.

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook`) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If the `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) and content is set then the return value is the size of the smallest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
2. If the `HLook` does not support scaling of content or no scaling is requested, *and* content is set then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If no content is available but a default preferred size has been set using `setDefaultSize(Dimension)` has been called to set then the return value is this value (as obtained with `getDefaultSize()`) plus any additional dimensions that the `HLook` requires for border decoration etc.
4. If there is no content or default size set then the return value is an implementation-specific minimum size plus any additional dimensions that the `HLook` requires for border decoration etc.

**Specified By:**

`getMinimumSize(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating this `HLook` minimum size.

**See Also:**

`getMinimumSize()`

---

**getPreferredSize(HVisible)**

```
public java.awt.Dimension getPreferredSize(HVisible hvisible)
```

Gets the preferred size of the `HVisible` component when drawn with this `HLook`.

This size may be determined in several ways depending on the information available to the look. These steps are performed in order and the first available result is returned. For the purposes of this algorithm `HLook` classes that do not use content (e.g. `HRangeLook`) are treated as if no content was present.

The extra space required for border decoration can be determined from the `getInsets(HVisible)` method.

1. If a default preferred size has been set for this `HVisible` (using `setDefaultSize(Dimension)`) then the return value is this size (obtained with `getDefaultSize()`) plus any additional dimensions that the `HLook` requires for border decoration etc.
2. If this `HLook` does not support scaling of content or no scaling is requested, and content is present then the return value is the size of the largest piece of content plus any additional dimensions that the `HLook` requires for border decoration etc.
3. If this `HLook` supports the scaling of its content (e.g. an `HGraphicLook`) and content is set then the

return value is the current size of the `HVisible` as returned by `getSize` ).

4. If there is no content and no default size set then the return value is the current size of the `HVisible` as returned by `getSize` ).

**Specified By:**

`getPreferredSize(HVisible)` in interface `HLook`

**Parameters:**

`hvisible` - `HVisible` to which this `HLook` is attached.

**Returns:**

A dimension object indicating the preferred size of the `HVisible` when drawn with this `HLook` .

**See Also:**

`getPreferredSize()` , `setDefaultSize(Dimension)`

### **isOpaque(HVisible)**

```
public boolean isOpaque(HVisible visible)
```

Returns true if the entire painted area of the `HVisible` when using this look is fully opaque, i.e. the `showLook(Graphics, HVisible, int)` method guarantees that all pixels are painted in an opaque `Color`.

The default value is implementation specific and depends on the background painting mode of the given `HVisible` . The consequences of an invalid overridden value are implementation specific.

**Specified By:**

`isOpaque(HVisible)` in interface `HLook`

**Parameters:**

`visible` - the visible to test

**Returns:**

true if all the pixels with the `java.awt.Component#getBounds` method of an `HVisible` using this look are fully opaque, i.e. the `showLook(Graphics, HVisible, int)` method guarantees that all pixels are painted in an opaque `Color`, otherwise false.

### **showLook(Graphics, HVisible, int)**

```
public void showLook(java.awt.Graphics g, HVisible visible, int state)
```

The `showLook(Graphics, HVisible, int)` method is responsible for repainting the entire `HVisible` component, (including any content set on the component, and the component background), subject to the `clipRect` of the `Graphics` object passed to it.

The `showLook(Graphics, HVisible, int)` method should not modify the `clipRect` of the `Graphics` object that is passed to it.

For looks which draw content (e.g. `HTextLook` , `HGraphicLook` and `HAnimateLook` ), if no content is associated with the component, the `showLook(Graphics, HVisible, int)` method paints the component with its current background `Color` according to the `setBackgroundMode(int)` method of `HVisible` and draws any (implementation-specific) borders. Note that by default the background mode is set so as to **not** paint a background. Furthermore on platforms which support transparent colors the background `Color` may be partially or completely transparent.

Any resources **explicitly** associated with an `HLook` should be loaded by the `HLook` during its creation, etc., or via its `setXXX()` methods. Note that the "standard" looks don't load content by default.

This method is called from the `paint(Graphics)` method of `HVisible` and must never be called from elsewhere. Components wishing to redraw themselves should call their repaint method in the usual way.

**Specified By:**

```
showLook(Graphics, HVisible, int) in interface HLook
```

**Parameters:**

`g` - the graphics context.

`visible` - the visible.

`state` - the state parameter indicates the state of the visible, allowing the look to render the appropriate content for that state. Note that some components (e.g. `HStaticRange`, `HRange`, `HRangeValue`) do not use state-based content).

---

**widgetChanged(HVisible, HChangeData[])**

```
public void widgetChanged(HVisible visible, HChangeData[] changes)
```

Called by the `HVisible` whenever its content, state, or any other data changes. See the class description of `HVisible` for more information about the `changes` parameter.

The implementation of this method should work out which graphical areas of the `HVisible` have changed and make any relevant calls to trigger the repainting of those areas.

A minimum implementation of this method could simply call

```
visible.repaint()
```

**Specified By:**

```
widgetChanged(HVisible, HChangeData[] ) in interface HLook
```

**Parameters:**

`visible` - the `HVisible` which has changed

`changes` - an array containing hint data and associated hint objects. If this argument is `null` a full repaint will be triggered.

# org.havi.ui

# HTextValue

## Syntax

```
public interface HTextValue extends HNavigable, HKeyboardInputPreferred
```

## All Superinterfaces:

```
HKeyboardInputPreferred, HNavigable, HNavigationInputPreferred
```

## All Known Implementing Classes:

```
HSinglelineEntry
```

## Description

This interface is implemented by all HAVi UI components which have have editable text content (e.g. a text entry control).

### Event Behavior

Subclasses of `HComponent` which implement `HTextValue` must respond to `HFocusEvent`, `HKeyEvent` and `HTextEvent` events.

Applications should assume that classes which implement `HTextValue` can generate events of the types `HFocusEvent` and `HTextEvent` in response to other types of input event.

An application may add one or more `HTextListener` listeners to the component. The `textChanged(HTextEvent)` method of the `HTextListener` is invoked whenever the text in the `HTextValue` is changed, and the `caretMoved(HTextEvent)` method of the `HTextListener` is invoked whenever the text caret position is altered.

HAVi text events are discussed in detail in the `HKeyboardInputPreferred` interface description.

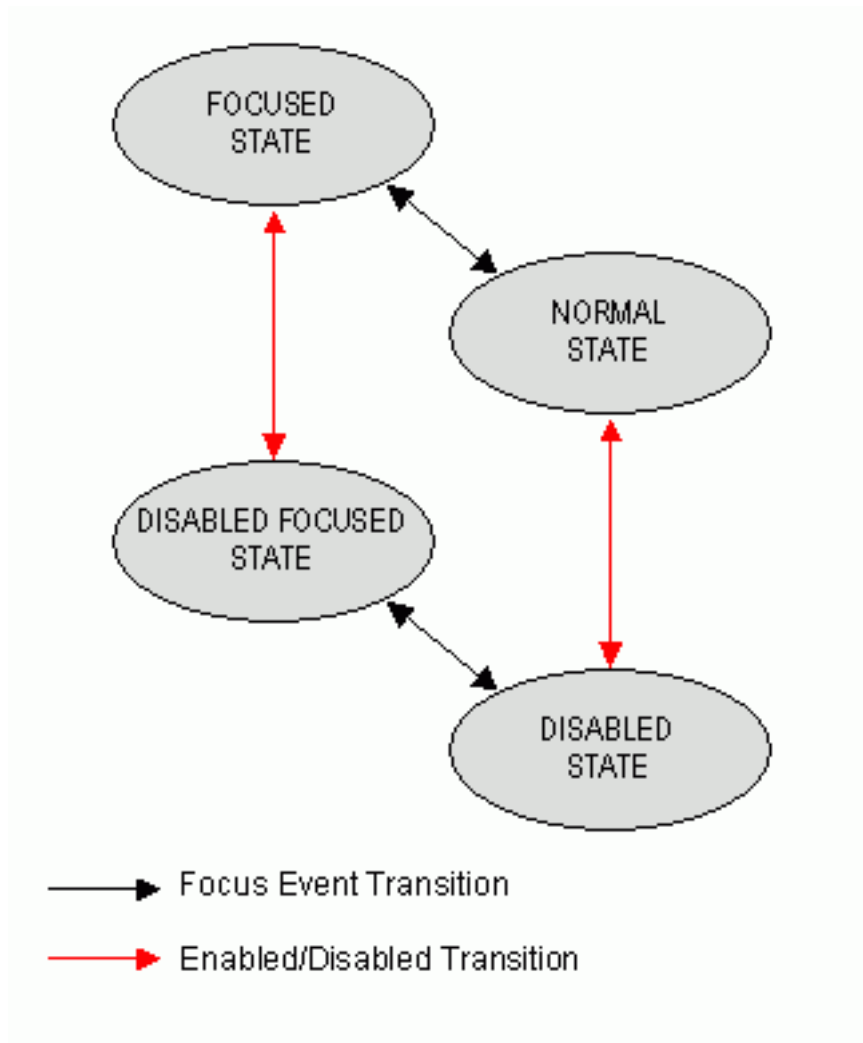
### Interaction States

The following interaction states are valid for this `HTextValue` component:

- `NORMAL_STATE`
- `FOCUSED_STATE`
- `DISABLED_STATE`
- `DISABLED_FOCUSED_STATE`

The state machine diagram below shows the valid state transitions for an `HTextValue` component.





#### Platform Classes

The following HAVi platform classes implement or inherit the `HTextValue` interface. These classes shall all generate both `HFocusEvent` and `HTextEvent` events in addition to any other events specified in the respective class descriptions.

- `HSinglelineEntry`
- `HMultilineEntry`

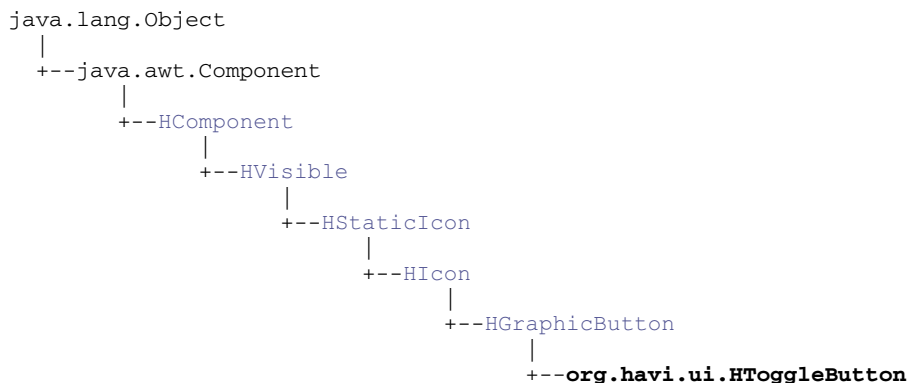
#### See Also:

`HNavigable`, `HKeyboardInputPreferred`, `HTextEvent`, `HTextListener`

# org.havi.ui HToggleButton

## Syntax

```
public class HToggleButton extends HGraphicButton implements HSwitchable
```



## All Implemented Interfaces:

`HActionable`, `HActionInputPreferred`, `HMatteLayer`, `HNavigable`,  
`HNavigationInputPreferred`, `HNoInputPreferred`, `HState`, `HSwitchable`,  
`java.awt.image.ImageObserver`, `java.awt.MenuContainer`, `java.io.Serializable`, `TestOpacity`

## Description

The `HToggleButton` is a user interface component representing a "check box", or with the support of the `HToggleGroup` class, "radio buttons". It displays static read-only graphical content. This component can be navigated to, i.e. it can have the input focus, and it can be actioned as defined by the `HSwitchable` interface. This means that the interaction state persists after `HActionEvent` event processing is complete.

The current switchable state can be manipulated using `setSwitchableState(boolean)` and `getSwitchableState()`

By default it uses the `HGraphicLook` class to render itself. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
width	width of this component in pixels (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>

height	height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
image	The image to be used as the content for every state of this component.	null	setGraphicContent (Image, int)	getGraphicContent (int)
imageNormal	The image to be used as the content for the <code>NORMAL_STATE</code> state of this component.	null	setGraphicContent (Image, int)	getGraphicContent (int)
imageFocus	The image to be used as the content for the focused states of this component.	null	setGraphicContent (Image, int)	getGraphicContent (int)
imageAction	The image to be used as the content for the actioned states of this component.	null	setGraphicContent (Image, int)	getGraphicContent (int)
imageNormal-Actioned	The image to be used as the content for the actioned without focus states of this component.	null	setGraphicContent (Image, int)	getGraphicContent (int)
state	The switchable state of this <code>HToggleButton</code> .	false	setSwitchableState (boolean)	getSwitchableState ()
group	The <code>HToggleGroup</code> with which to associate this <code>HToggleButton</code> .	null	setToggleGroup (HToggleGroup)	getToggleGroup ()

#### Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte ( <code>HMatte</code> ).	none (i.e. <code>getMatte()</code> returns null)	<code>setMatte (HMatte)</code>	<code>getMatte ()</code>
The text layout manager responsible for text formatting.	An <code>HDefaultTextLayoutManager</code> object.	<code>setTextLayoutManager (HTextLayoutManager)</code>	<code>getTextLayoutManager ()</code>
The background painting mode	<code>NO_BACKGROUND_FILL</code>	<code>setBackgroundMode (int)</code>	<code>getBackgroundMode ()</code>
The default preferred size	not set (i.e. null) unless specified by width and height parameters	<code>setDefaultSize (Dimension)</code>	<code>getDefaultSize ()</code>
The horizontal content alignment	<code>HALIGN_CENTER</code>	<code>setHorizontalAlignment (int)</code>	<code>getHorizontalAlignment ()</code>
The vertical content alignment	<code>VALIGN_CENTER</code>	<code>setVerticalAlignment (int)</code>	<code>getVerticalAlignment ()</code>

The content scaling mode	RESIZE_NONE	setResizeMode (int)	getResizeMode ()
The default "look" for this class.	A platform specific HGraphicLook	setDefaultLook (HGraphicLook)	getDefaultLook ()
The "look" for this object.	The HGraphicLook returned from HToggleButton.getDefaultLook when this object was created.	setLook (HLook)	getLook ()
The gain focus sound.	null	setGainFocusSound (HSound)	getGainFocusSound ()
The lose focus sound.	null	setLoseFocusSound (HSound)	getLoseFocusSound ()
The action sound.	null	setActionSound (HSound)	getActionSound ()
The unset action sound.	null	setUnsetActionSound (HSound)	getUnsetActionSound ()

**See Also:**

HStaticIcon, HIcon, HNavigable, HActionable, HSwitchable

## Constructors

---

**HToggleButton()**

```
public HToggleButton()
```

Creates an `HToggleButton` object. See the class description for details of constructor parameters and default values.

---

**HToggleButton(Image)**

```
public HToggleButton(java.awt.Image image)
```

Creates an `HToggleButton` object. See the class description for details of constructor parameters and default values.

---

**HToggleButton(Image, boolean, HToggleGroup)**

```
public HToggleButton(java.awt.Image image, boolean state, HToggleGroup group)
```

Creates an `HToggleButton` object. See the class description for details of constructor parameters and default values.

---

**HToggleButton(Image, Image, Image, Image, boolean)**

```
public HToggleButton(java.awt.Image imageNormal, java.awt.Image imageFocused,  
    java.awt.Image imageActioned, java.awt.Image imageNormalActioned,  
    boolean state)
```

Creates an `HToggleButton` object. See the class description for details of constructor parameters and default values.

---

**HToggleButton(Image, Image, Image, Image, boolean, HToggleGroup)**

```
public HToggleButton(java.awt.Image imageNormal, java.awt.Image imageFocused,  
    java.awt.Image imageActioned, java.awt.Image imageNormalActioned,  
    boolean state, HToggleGroup group)
```

Creates an `HToggleButton` object. See the class description for details of constructor parameters and default values.

---

**HToggleButton(Image, Image, Image, Image, int, int, int, int, boolean)**

```
public HToggleButton(java.awt.Image imageNormal, java.awt.Image imageFocused,  
    java.awt.Image imageActioned, java.awt.Image imageNormalActioned, int x,  
    int y, int width, int height, boolean state)
```

Creates an `HToggleButton` object. See the class description for details of constructor parameters and default values.

---

**HToggleButton(Image, Image, Image, Image, int, int, int, int, boolean, HToggleGroup)**

```
public HToggleButton(java.awt.Image imageNormal, java.awt.Image imageFocused,  
    java.awt.Image imageActioned, java.awt.Image imageNormalActioned, int x,  
    int y, int width, int height, boolean state, HToggleGroup group)
```

Creates an `HToggleButton` object. See the class description for details of constructor parameters and default values.

---

**HToggleButton(Image, int, int, int, int)**

```
public HToggleButton(java.awt.Image image, int x, int y, int width, int height)
```

Creates an `HToggleButton` object. See the class description for details of constructor parameters and default values.

---

**HToggleButton(Image, int, int, int, int, boolean)**

```
public HToggleButton(java.awt.Image image, int x, int y, int width, int height,  
    boolean state)
```

Creates an `HToggleButton` object. See the class description for details of constructor parameters and default values.

---

**HToggleButton(Image, int, int, int, int, boolean, HToggleGroup)**

```
public HToggleButton(java.awt.Image image, int x, int y, int width, int height,  
    boolean state, HToggleGroup group)
```

Creates an `HToggleButton` object. See the class description for details of constructor parameters and default values.

## Methods

---

### addActionListener(HActionListener)

```
public void addActionListener(HActionListener l)
```

Adds the specified `HActionListener` to receive `HActionEvent` events sent from this `HToggleButton`. If the listener has already been added further calls will add further references to the listener, which will then receive multiple copies of a single event.

**Specified By:**

`addActionListener(HActionListener)` in interface `HActionable`

**Overrides:**

`addActionListener(HActionListener)` in class `HGraphicButton`

**Parameters:**

`l` - the `HActionListener`.

---

### getActionCommand()

```
public java.lang.String getActionCommand()
```

Gets the command name for the `HActionEvent` event fired by this `HToggleButton`.

**Specified By:**

`getActionCommand()` in interface `HActionable`

**Overrides:**

`getActionCommand()` in class `HGraphicButton`

**Returns:**

A `String` representing the command name of the `HActionEvent` fired by this `HToggleButton`.

**See Also:**

`getActionCommand()`

---

### getActionSound()

```
public HSound getActionSound()
```

Associate a sound to be played when the interaction state of the `HToggleButton` makes the following transitions:

- `_NORMAL_STATE` to `_ACTIONED_STATE`
- `_FOCUSED_STATE` to `_ACTIONED_FOCUSED_STATE`

**Specified By:**

`getActionSound()` in interface `HActionable`

**Overrides:**

`getActionSound()` in class `HGraphicButton`

**Returns:**

The sound played when the component is actioned.

---

### getDefaultLook()

```
public static HGraphicLook getDefaultLook()
```

Returns the currently set default `HLook` for `HToggleButton` components.

---

**Returns:**

The `HLook` that is used by default when creating a new `HToggleButton` component.

---

**getGainFocusSound()**

```
public HSound getGainFocusSound()
```

Get the sound associated with the gain focus event.

**Specified By:**

```
getGainFocusSound() in interface HNavigable
```

**Overrides:**

```
getGainFocusSound() in class HGraphicButton
```

**Returns:**

The sound played when the component gains focus. If no sound is associated with gaining focus, then null shall be returned.

---

**getLoseFocusSound()**

```
public HSound getLoseFocusSound()
```

Get the sound associated with the lost focus event.

**Specified By:**

```
getLoseFocusSound() in interface HNavigable
```

**Overrides:**

```
getLoseFocusSound() in class HGraphicButton
```

**Returns:**

The sound played when the component loses focus. If no sound is associated with losing focus, then null shall be returned.

---

**getMove(int)**

```
public HNavigable getMove(int keyCode)
```

Provides the `HNavigable` object that is navigated to when a particular key is pressed.

**Specified By:**

```
getMove(int) in interface HNavigable
```

**Overrides:**

```
getMove(int) in class HGraphicButton
```

**Parameters:**

`keyCode` - The key code of the pressed key.

**Returns:**

Returns the `HNavigable` object, or if no `HNavigable` is associated with the `keyCode` then returns null.

---

**getNavigationKeys()**

```
public int[] getNavigationKeys()
```

Retrieve the set of key codes which this component maps to navigation targets.

**Specified By:**

```
getNavigationKeys() in interface HNavigationInputPreferred
```

**Overrides:**

`getNavigationKeys()` in class `HGraphicButton`

**Returns:**

an array of key codes, or `null` if no navigation targets are set on this component.

---

**getSwitchableState()**

```
public boolean getSwitchableState()
```

Returns the current switchable state of this `HSwitchable`.

**Specified By:**

`getSwitchableState()` in interface `HSwitchable`

**Returns:**

the current switchable state of this `HSwitchable`.

---

**getToggleGroup()**

```
public HToggleGroup getToggleGroup()
```

Gets the `HToggleGroup` the `HToggleButton` is associated with.

**Returns:**

The `HToggleGroup` the `HToggleButton` is associated with, or `null` if the `HToggleButton` is not associated with an `HToggleGroup`.

---

**getUnsetActionSound()**

```
public HSound getUnsetActionSound()
```

Get the sound to be played when the interaction state of the `HSwitchable` makes the following transitions:

- `_ACTIONED_STATE` to `_NORMAL_STATE`
- `_ACTIONED_FOCUSED_STATE` to `_FOCUSED_STATE`

**Specified By:**

`getUnsetActionSound()` in interface `HSwitchable`

**Returns:**

the sound to be played when the `HSwitchable` transitions from an actioned state.

---

**isSelected()**

```
public boolean isSelected()
```

Indicates if this component has focus.

**Specified By:**

`isSelected()` in interface `HNavigable`

**Overrides:**

`isSelected()` in class `HGraphicButton`

**Returns:**

`true` if the component has focus, otherwise returns `false`.

---

**processHActionEvent(HActionEvent)**

```
public void processHActionEvent(HActionEvent evt)
```

Process an `HActionEvent` sent to this `HToggleButton`.

---



**Specified By:**

`processHActionEvent(HActionEvent)` in interface `HActionInputPreferred`

**Overrides:**

`processHActionEvent(HActionEvent)` in class `HGraphicButton`

**Parameters:**

`evt` - the `HActionEvent` to process.

---

**processHFocusEvent(HFocusEvent)**

```
public void processHFocusEvent(HFocusEvent evt)
```

Process an `HFocusEvent` sent to this `HToggleButton`.

**Specified By:**

`processHFocusEvent(HFocusEvent)` in interface `HNavigationInputPreferred`

**Overrides:**

`processHFocusEvent(HFocusEvent)` in class `HGraphicButton`

**Parameters:**

`evt` - the `HFocusEvent` to process.

---

**removeHActionListener(HActionListener)**

```
public void removeHActionListener(HActionListener l)
```

Removes the specified `HActionListener` so that it no longer receives `HActionEvent` events from this `HToggleButton`. If the specified listener is not registered, the method has no effect. If multiple references to a single listener have been registered it should be noted that this method will only remove one reference per call.

**Specified By:**

`removeHActionListener(HActionListener)` in interface `HActionable`

**Overrides:**

`removeHActionListener(HActionListener)` in class `HGraphicButton`

**Parameters:**

`l` - the `HActionListener`.

---

**removeToggleGroup()**

```
public void removeToggleGroup()
```

Removes the button from the toggle group that it has been added to. This method does nothing if the button had not been previously added to an `HToggleGroup`.

---

**setActionCommand(String)**

```
public void setActionCommand(java.lang.String command)
```

Sets the command name for the `HActionEvent` event fired by this `HToggleButton`.

**Specified By:**

`setActionCommand(String)` in interface `HActionable`

**Overrides:**

`setActionCommand(String)` in class `HGraphicButton`

**Parameters:**

`command` - a `String` used to set the action command.

---

**See Also:**

`getActionCommand()`

**setActionSound(HSound)**

```
public void setActionSound(HSound sound)
```

Associate a sound to be played when the interaction state of the `HToggleButton` makes the following transitions:

- `_NORMAL_STATE` to `_ACTIONED_STATE`
- `_FOCUSED_STATE` to `_ACTIONED_FOCUSED_STATE`

**Specified By:**

`setActionSound(HSound)` in interface `HActionable`

**Overrides:**

`setActionSound(HSound)` in class `HGraphicButton`

**Parameters:**

`sound` - the sound to be played, when the component is actioned. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

**setDefaultLook(HGraphicLook)**

```
public static void setDefaultLook(HGraphicLook hlook)
```

Sets the default `HLook` for further `HToggleButton` components.

**Parameters:**

`hlook` - The `HLook` that will be used by default when creating a new `HToggleButton` component. Note that this parameter may be null, in which case newly created components shall not draw themselves until a non-null look is set using the `setLook(HLook)` method.

**setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)**

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
    HNavigable right)
```

Set the focus control for an `HNavigable` component. Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is a convenience function for application programmers where a standard up, down, left and right focus traversal between components is required.

Note `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

Note that this API does not prevent the creation of "isolated" `HNavigable` components --- authors should endeavor to avoid confusing the user.

**Specified By:**

`setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` in interface `HNavigable`

**Overrides:**

`setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` in class `HGraphicButton`

**Parameters:**

`up` - The `HNavigable` component to move to, when the user generates a `VK_UP` KeyEvent. If there is no `HNavigable` component to move "up" to, then null should be specified.

`down` - The `HNavigable` component to move to, when the user generates a `VK_DOWN` `KeyEvent`. If there is no `HNavigable` component to move "down" to, then null should be specified.

`left` - The `HNavigable` component to move to, when the user generates a `VK_LEFT` `KeyEvent`. If there is no `HNavigable` component to move "left" to, then null should be specified.

`right` - The `HNavigable` component to move to, when the user generates a `VK_RIGHT` `KeyEvent`. If there is no `HNavigable` component to move "right" to, then null should be specified.

### **setGainFocusSound(HSound)**

```
public void setGainFocusSound(HSound sound)
```

Associate a sound with gaining focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_GAINED`. This sound will start to be played when an object implementing this interface gains focus. It is not guaranteed to be played to completion. If the object implementing this interface loses focus before the audio completes playing, the audio will be truncated. Applications wishing to ensure the audio is always played to completion must implement special logic to slow down the focus transitions.

By default, an `HNavigable` object does not have any gain focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

#### **Specified By:**

`setGainFocusSound(HSound)` in interface `HNavigable`

#### **Overrides:**

`setGainFocusSound(HSound)` in class `HGraphicButton`

#### **Parameters:**

`sound` - the sound to be played, when the component gains focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

### **setLoseFocusSound(HSound)**

```
public void setLoseFocusSound(HSound sound)
```

Associate a sound with losing focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_LOST`. This sound will start to be played when an object implementing this interface loses focus. It is not guaranteed to be played to completion. It is implementation dependent whether and when this sound will be truncated by any gain focus sound played by the next object to gain focus.

By default, an `HNavigable` object does not have any lose focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

#### **Specified By:**

`setLoseFocusSound(HSound)` in interface `HNavigable`

#### **Overrides:**

`setLoseFocusSound(HSound)` in class `HGraphicButton`

#### **Parameters:**

`sound` - the sound to be played, when the component loses focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

### **setMove(int, HNavigable)**

```
public void setMove(int keyCode, HNavigable target)
```

Defines the navigation path from the current `HNavigable` to another `HNavigable` when a particular key is pressed.

Note that `setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)` is equivalent to multiple calls to `setMove(int, HNavigable)`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

**Specified By:**

`setMove(int, HNavigable)` in interface `HNavigable`

**Overrides:**

`setMove(int, HNavigable)` in class `HGraphicButton`

**Parameters:**

`keyCode` - The key code of the pressed key. Any numerical keycode is allowed, but the platform may not be able to generate all keycodes. Application authors should only use keys for which `HRcCapabilities.isSupported()` or `HKeyCapabilities.isSupported()` returns true.

`target` - The target `HNavigable` object that should be navigated to. If a target is to be removed from a particular navigation path, then `null` should be specified.

### **setSwitchableState(boolean)**

```
public void setSwitchableState(boolean state)
```

Sets the current state of the button. Note that `ActionListeners` are only called when an `ACTION_PERFORMED` event is received, or if they are called directly, e.g. via `processActionEvent`, they are not called by `setSwitchableState(boolean)`.

**Specified By:**

`setSwitchableState(boolean)` in interface `HSwitchable`

### **setToggleGroup(HToggleGroup)**

```
public void setToggleGroup(HToggleGroup group)
```

Associates the `HToggleButton` with an `HToggleGroup`. If this `HToggleButton` is already in a different `HToggleGroup`, it is first taken out of that group.

**Parameters:**

`group` - The `HToggleGroup` the `HToggleButton` is to be associated with.

### **setUnsetActionSound(HSound)**

```
public void setUnsetActionSound(HSound sound)
```

Associate a sound to be played when the interaction state of the `HSwitchable` makes the following transitions:

- `_ACTIONED_STATE` to `_NORMAL_STATE`
- `_ACTIONED_FOCUSED_STATE` to `_FOCUSED_STATE`

**Specified By:**

`setUnsetActionSound(HSound)` in interface `HSwitchable`

**Parameters:**

`sound` - a sound to be played when the `HSwitchable` transitions from an actioned state. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

# org.havi.ui

# HToggleGroup

## Syntax

```
public class HToggleGroup

java.lang.Object
|
+--org.havi.ui.HToggleGroup
```

## Description

`HToggleButton` within the same `HToggleGroup` will behave so that a maximum of one `HToggleButton` has switchable state `true`, as returned by `getSwitchableState()`, so as to achieve a "radio button" effect. When an `HToggleButton` is acted upon to change its switchable state to `true`, then if any other `HToggleButton` within the `HToggleGroup` currently has switchable state `true`, it will have its switchable state set to `false`. Similarly, if an `HToggleButton` is added which has switchable state `true`, then any current `HToggleButton` within the `HToggleGroup` with switchable state `true`, shall have its switchable state modified to `false`.

If the forced selection mode for the `HToggleGroup` is set via a call to `setForcedSelection(true)` then there will always be one `HToggleButton` selected (i.e. with switchable state `true`), and if necessary the `HToggleGroup` will automatically force a selection to ensure this. If forced selection mode is not set it is valid for there to be no selection, i.e. all `HToggleButton` may have switchable state `false`. By default forced selection mode is not set.

Note that when an `HToggleButton` has switchable state `true` this implies that the interaction state as returned by `getInteractionState()` will be either the `ACTIONED_STATE` or `ACTIONED_FOCUSED_STATE` state.

Similarly, a switchable state of `false` implies that the interaction state is any other state for which the `ACTIONED_STATE_BIT` is not set. See the `HSwitchable` class description for more information about the valid interaction states. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Enable or disable buttons in group	<code>enable</code>	<code>setEnabled(boolean)</code>	<code>isEnabled()</code>
Forced selection mode.	<code>false</code>	<code>setForcedSelection(boolean)</code>	<code>getForcedSelection()</code>

## Constructors

---

## HToggleGroup()

```
public HToggleGroup()
```

Creates a new version instance of an `HToggleGroup`

## Methods

---

### add(HToggleButton)

```
protected void add(HToggleButton button)
```

Add the specified `HToggleButton` to this `HToggleGroup`. The `HToggleButton` will be enabled or disabled as necessary to match the current state of the `HToggleGroup` as determined by the `isEnabled()` method.

If the `HToggleGroup` is empty and forced selection mode is active then the `HToggleButton` will automatically have its switchable state set to `true`. Note that any `ActionListeners` registered with the `HToggleButton` are not called as a result of this operation.

This method is protected to allow the platform to override it in subclasses of `HToggleGroup`. It is not intended for use by the application and conformant applications shall not use this method.

**Parameters:**

`button` - the `HToggleButton` to add to the `HToggleGroup`.

---

### getCurrent()

```
public HToggleButton getCurrent()
```

Returns the `HToggleButton` from this `HToggleGroup` which has state `true`, or null otherwise, for example, if there are no `HToggleButton` associated with this `HToggleGroup`, or if all `HToggleButton` within this `HToggleGroup` have state `false`.

**Returns:**

the currently selected `HToggleButton` or null if no such `HToggleButton` exists.

---

### getForcedSelection()

```
public boolean getForcedSelection()
```

Return the current forced selection mode of the group.

**Returns:**

the current forced selection mode. If this value is `true` the group is forced to always have one `HToggleButton` selected (i.e. with switchable state `true`). Otherwise, the group may have either one or zero `HToggleButton` selected.

---

### isEnabled()

```
public boolean isEnabled()
```

Determines whether the `HToggleGroup` is enabled. `HToggleGroup` are enabled initially by default. A group may be enabled or disabled by calling its `setEnabled` method.

**Returns:**

`true` if the component is enabled; `false` otherwise.

**See Also:**

```
setEnabled(boolean)
```

---

### **remove(HToggleButton)**

```
protected void remove(HToggleButton button)
```

Remove the specified `HToggleButton` to this `HToggleGroup`. If `button` is not part of this `HToggleGroup` this method throws a `java.lang.IllegalArgumentException`.

If the `HToggleButton` is the currently selected button in this group and forced selection mode is set the first remaining `HToggleButton` will automatically have its switchable state set to true. Note that any `ActionListeners` registered with the `HToggleButton` are not called as a result of this operation.

This method is protected to allow the platform to override it in subclasses of `HToggleGroup`. It is not intended for use by the application and conformant applications shall not use this method.

**Parameters:**

`button` - the `HToggleButton` to remove from the `HToggleGroup`.

**Throws:**

`java.lang.IllegalArgumentException` - if `button` is not a member of this `HToggleGroup`.

---

### **setCurrent(HToggleButton)**

```
public void setCurrent(HToggleButton selection)
```

If the specified `HToggleButton` is a member of this `HToggleGroup`, then it is selected, its state is set to true and consequently any other `HToggleButton` within the `HToggleGroup` will have their states set to false.

If the specified `HToggleButton` is not a member of this `HToggleGroup`, then no actions are performed.

**Parameters:**

`selection` - the `HToggleButton` to be set as the currently selected item within the `HToggleGroup`.

---

### **setEnabled(boolean)**

```
public void setEnabled(boolean enable)
```

Enables or disables the group, depending on the value of the parameter `enable`. An enabled group's `HToggleButton` can respond to user input and generate events. An `HToggleGroup` is initially enabled by default.

Enabling or disabling an `HToggleGroup` enables or disables all the `HToggleButton` components in the group by calling their `setEnabled` methods.

Whether or not a group is enabled does not affect the adding or removing of `HToggleButton` from that group.

**Parameters:**

`enable` - true to enable all the `HToggleButton` components in the group, false to disable them.

**See Also:**

`isEnabled()`

---

---

**setForcedSelection(boolean)**

```
public void setForcedSelection(boolean forceSelection)
```

Set the forced selection mode of the group. If `forceSelection` is `true` and no `HToggleButton` is currently selected in the group the first `HToggleButton` automatically has its switchable state set to `true`. If no `HToggleButton` components have been added to this group there will be no visual indication of the mode change.

**Parameters:**

`forceSelection` - if this parameter is `true` the group is forced to always have one `HToggleButton` selected (i.e. with switchable state `true`). Otherwise, the group may have either one or zero `HToggleButton` selected.



# org.havi.ui HUIException

## Syntax

```
public class HUIException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--org.havi.ui.HUIException
```

## Direct Known Subclasses:

`HInvalidLookException`, `HMatteException`

## All Implemented Interfaces:

`java.io.Serializable`

## Description

`HUIException` is a generic exception that indicates that the desired user-interface mechanism cannot be performed for some reason. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
message	Message explaining why the exception was thrown	null	-	<code>java.lang.Throwable#getMessage</code>

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Constructors

### HUIException()

```
public HUIException()
```

Creates an `HUIException` object. See the class description for details of constructor parameters and default values.

---

**HUIException(String)**

```
public HUIException(java.lang.String message)
```

Creates an HUIException object with a specified reason string.

**Parameters:**

message - the reason why the exception was raised

# org.havi.ui

# HVersion

## Syntax

```
public interface HVersion
```

## Description

The `HVersion` interface defines some versioning constants that are accessible by using the `java.lang.System` method `getProperty`, with the appropriate property name.

Note that it is a valid implementation to return empty strings for the implementation, vendor and name strings.

## Fields

---

### HAVI\_IMPLEMENTATION\_NAME

```
public static final java.lang.String HAVI_IMPLEMENTATION_NAME
```

A string constant describing the HAVi implementation name, as returned via `java.lang.System.getProperty(havi.implementation.name)`.

---

### HAVI\_IMPLEMENTATION\_VENDOR

```
public static final java.lang.String HAVI_IMPLEMENTATION_VENDOR
```

A string constant describing the HAVi implementation vendor, as returned via `java.lang.System.getProperty(havi.implementation.vendor)`.

---

### HAVI\_IMPLEMENTATION\_VERSION

```
public static final java.lang.String HAVI_IMPLEMENTATION_VERSION
```

A string constant describing the HAVi implementation version, as returned via `java.lang.System.getProperty(havi.implementation.version)`.

---

### HAVI\_SPECIFICATION\_NAME

```
public static final java.lang.String HAVI_SPECIFICATION_NAME
```

A string constant describing the HAVi specification name, as returned via `java.lang.System.getProperty(havi.specification.name)`.

---

### HAVI\_SPECIFICATION\_VENDOR

```
public static final java.lang.String HAVI_SPECIFICATION_VENDOR
```

A string constant describing the HAVi specification vendor, as returned via `java.lang.System.getProperty(havi.specification.vendor)`.

---

**HAVI\_SPECIFICATION\_VERSION**

```
public static final java.lang.String HAVI_SPECIFICATION_VERSION
```

A string constant describing the HAVi specification version, as returned via `java.lang.System.getProperty(havi.specification.version)`.

# org.havi.ui HVideoComponent

## Syntax

public class HVideoComponent extends HComponent

```

java.lang.Object
|
+--java.awt.Component
    |
    +--HComponent
        |
        +--org.havi.ui.HVideoComponent
  
```

## All Implemented Interfaces:

HMatteLayer, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, TestOpacity

## Description

`HVideoComponent` is an opaque class encapsulating the presentation of a video source *within* an application, i.e. contained within a conventional AWT hierarchy.

An `HVideoComponent` obeys all conventional `java.awt.Component` semantics, including being clipped by its parent container, etc. An `HVideoComponent` also obeys all `HComponent` semantics including Z-ordering, etc. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Constructors

### HVideoComponent()

```
protected HVideoComponent()
```

It is not intended that applications should directly construct `HVideoComponent` objects.

Creates an `HVideoComponent` object. See the class description for details of constructor parameters and default values.

## Methods

---

### **addOnScreenLocationModifiedListener(HScreenLocationModifiedListener)**

```
public void addOnScreenLocationModifiedListener(HScreenLocationModifiedListener slml)
```

Register a listener to determine if the Component's on-screen location is modified - irrespective of its relative location to its parent Container. If the listener has already been added further calls will add further references to the listener, which will then receive multiple copies of a single event.

**Parameters:**

`slml` - listener to be notified when the on-screen location of the component is modified.

---

### **getVideoDevice()**

```
public HVideoDevice getVideoDevice()
```

Returns the `HVideoDevice` that this `HVideoComponent` is associated with.

**Returns:**

the `HVideoDevice` that this `HVideoComponent` is associated with, or null if this cannot be determined.

---

### **removeOnScreenLocationModifiedListener(HScreenLocationModifiedListener)**

```
public void removeOnScreenLocationModifiedListener(HScreenLocationModifiedListener slml)
```

Remove a listener that determines if the Component's on-screen location is modified - irrespective of its relative location to its parent Container. If the specified listener is not registered, the method has no effect. If multiple references to a single listener have been registered it should be noted that this method will only remove one reference per call.

**Parameters:**

`slml` - listener to be notified when the on-screen location of the component is modified.

# org.havi.ui HVideoConfigTemplate

## Syntax

```
public class HVideoConfigTemplate extends HScreenConfigTemplate
```

```
java.lang.Object
|
+--HScreenConfigTemplate
|
+--org.havi.ui.HVideoConfigTemplate
```

## Description

The `HVideoConfigTemplate` class is used to obtain a valid `HVideoConfiguration`. An application instantiates one of these objects and then sets all non-default attributes as desired. The object is then passed to the `getBestConfiguration(HVideoConfigTemplate)` method found in the `HVideoDevice` class. If possible, a valid `HVideoConfiguration` is returned which meets or exceeds the requirements set in the `HVideoConfigTemplate`.

This class may be subclassed to support additional properties of video configurations which may be requested by applications. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## See Also:

`HScreenConfigTemplate`, `HBackgroundConfigTemplate`, `HGraphicsConfigTemplate`

## Fields

### GRAPHICS\_MIXING

```
public static final int GRAPHICS_MIXING
```

A value for use in the preference field of the `setPreference(int, int)` and `getPreferencePriority(int)` methods in the `HVideoConfigTemplate` that indicates that the device configuration supports the display of graphics in addition to video streams. This display includes both configurations where the video pixels and graphics pixels are fully aligned (same size) as well as configurations where they are displayed together but where a more complex relationship exists between the two pixel coordinate spaces. The graphics configuration for mixing is specified as an `HGraphicsConfiguration`.

This preference is used by the platform as a constraint in selecting configurations. Templates generated by the platform and then returned to applications (e.g. from a `getConfigTemplate` method) shall not have this preference filled in by the platform.

## Constructors

---

### HVideoConfigTemplate()

```
public HVideoConfigTemplate()
```

Creates an `HVideoConfigTemplate` object. See the class description for details of constructor parameters and default values.

## Methods

---

### getPreferenceObject(int)

```
public java.lang.Object getPreferenceObject(int preference)
```

Return the preference object for the specified preference.

Instances of `HVideoConfigTemplate` which have not had this preference set shall return null for this object. Note that instances constructed by the platform and returned to applications are required to have all preferences (except where explicitly identified) set by the platform before it is returned.

**Overrides:**

```
getPreferenceObject(int) in class HScreenConfigTemplate
```

**Parameters:**

`preference` - the preference to be indicated. Valid values for an `HVideoConfigTemplate` are: `VIDEO_GRAPHICS_PIXEL_ALIGNED`, `PIXEL_ASPECT_RATIO`, `PIXEL_RESOLUTION`, `SCREEN_RECTANGLE`, and `GRAPHICS_MIXING`.

Subclasses may add further valid values. A `IllegalArgumentException` shall be thrown if the preference is not a valid value for this instance of `HVideoConfigTemplate`, or if `preference` does not have an associated value object.

**Returns:**

the preference object for the specified preference.

---

### getPreferencePriority(int)

```
public int getPreferencePriority(int preference)
```

Return the priority for the specified preference.

By default the preferences in a template returned from the system will have an `DONT_CARE` priority unless specified otherwise. Any configuration always satisfies these attributes.

**Overrides:**

```
getPreferencePriority(int) in class HScreenConfigTemplate
```

**Parameters:**

`preference` - the preference to be indicated. Valid values for an `HVideoConfigTemplate` are: `ZERO_BACKGROUND_IMPACT`, `ZERO_GRAPHICS_IMPACT`, `ZERO_VIDEO_IMPACT`, `INTERLACED_DISPLAY`, `FLICKER_FILTERING`, `VIDEO_GRAPHICS_PIXEL_ALIGNED`,



PIXEL\_ASPECT\_RATIO , PIXEL\_RESOLUTION , SCREEN\_RECTANGLE and GRAPHICS\_MIXING .

Subclasses may add further valid values. An `IllegalArgumentException` shall be thrown if the preference is not a valid value for this instance of `HVideoConfigTemplate`

**Returns:**

the priority for the specified preference.

### **isConfigSupported(HVideoConfiguration)**

```
public boolean isConfigSupported(HVideoConfiguration hvc)
```

Returns a boolean indicating whether or not the specified `HVideoConfiguration` can be used to create a video plane that supports the features set in this template.

**Parameters:**

`hvc` -- the `HVideoConfiguration` object to test against this template.

**Returns:**

true if this `HVideoConfiguration` object can be used to create a video plane that supports the features set in this template, false otherwise.

### **setPreference(int, Object, int)**

```
public void setPreference(int preference, java.lang.Object object, int priority)
```

Set the indicated preference (and associated value object) to have the specified priority. If the preference has been previously set, then the previous object and priority shall be overwritten.

Attributes that are not filled in in a template (through `setPreference(int, Object, int)` ), shall have the priority `DONT_CARE` . Any configuration always satisfies these attributes.

An application which wishes to remove a preference from an existing template (e.g. one generated by the platform) may call this method with null for the object parameter.

**Overrides:**

`setPreference(int, Object, int)` in class `HScreenConfigTemplate`

**Parameters:**

`preference` - the preference to be indicated. Valid values for an `HScreenConfigTemplate` are: `PIXEL_ASPECT_RATIO` , `PIXEL_RESOLUTION` , `SCREEN_RECTANGLE` , `VIDEO_GRAPHICS_PIXEL_ALIGNED` and `GRAPHICS_MIXING` .

Subclasses may add further valid values. An `IllegalArgumentException` shall be thrown if the preference is not a valid value for this instance of `HVideoConfigTemplate`

`object` - the Object associated with the given preference, or null.

`priority` - the priority of the preference. Valid values include: `REQUIRED` , `PREFERRED` , `DONT_CARE` , `PREFERRED_NOT` and `REQUIRED_NOT` .

If `priority` is not a valid priority as defined here a `java.lang.IllegalArgumentException` will be thrown.

# org.havi.ui HVideoConfiguration

## Syntax

public class HVideoConfiguration extends HScreenConfiguration

```
java.lang.Object
|
+--HScreenConfiguration
|
+--org.havi.ui.HVideoConfiguration
```

## Description

The `HVideoConfiguration` class describes the characteristics (settings) of an `HVideoDevice`. There can be many `HVideoConfiguration` objects associated with a single `HVideoDevice`. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## See Also:

`HVideoDevice`

## Constructors

### HVideoConfiguration()

```
protected HVideoConfiguration()
```

It is not intended that applications should directly construct `HVideoConfiguration` objects.

Creates an `HVideoConfiguration` object. See the class description for details of constructor parameters and default values.

## Methods

---

**getConfigTemplate()**

```
public HVideoConfigTemplate getConfigTemplate()
```

Returns an `HVideoConfigTemplate` object that describes and uniquely identifies this `HVideoConfiguration`.

Hence, the following sequence should return the original `HVideoConfiguration`.

```
HVideoDevice.getBestMatch(HVideoConfiguration .getConfigTemplate())
```

Features that are implemented in the `HVideoConfiguration` will return `REQUIRED` priority. Features that are not implemented in the `HVideoConfiguration` will return `REQUIRED_NOT` priority.

**Returns:**

an `HVideoConfigTemplate` object which both describes and uniquely identifies this `HVideoConfiguration`.

---

**getDevice()**

```
public HVideoDevice getDevice()
```

Returns the `HVideoDevice` associated with this `HVideoConfiguration`.

**Returns:**

the `HVideoDevice` object that is associated with this `HVideoConfiguration`,

# org.havi.ui HVideoDevice

## Syntax

```
public class HVideoDevice extends HScreenDevice
```

```
java.lang.Object
|
+--HScreenDevice
|
+--org.havi.ui.HVideoDevice
```

## All Implemented Interfaces:

org.davic.resources.ResourceProxy, org.davic.resources.ResourceServer

## Description

The `HVideoDevice` class describes the logical video devices which can contribute to the appearance of a particular screen. Each `HVideoDevice` has one or more `HVideoConfiguration` objects associated with it. These objects specify the different configurations (settings) in which the `HVideoDevice` can be used. This class represents the presentation only of video and does not provide for the selection of which video is to be presented. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## See Also:

`HVideoConfiguration`, `HScreenDevice`, `HScreen`

## Fields

### NOT\_CONTRIBUTING

```
public static final HVideoConfiguration NOT_CONTRIBUTING
```

Constant indicating that a video device is not contributing to the output of its `HScreen`. This shall only be used in implementations where an `HVideoDevice` shares an underlying MPEG decoder with an `HBackgroundDevice` when the `HBackgroundDevice` is in an `HStillImageBackgroundConfiguration`. In such implementations, this configuration shall be returned from the `getCurrentConfiguration()` method when the underlying video decoder is being used for displaying `HBackgroundImage`. It shall not be returned from any other method. The

`setVideoConfiguration(HVideoConfiguration)` method shall consider this to be an invalid configuration for this device and fail as specified.

On implementations where `getCurrentConfiguration()` never returns this value, `NOT_CONTRIBUTING` shall be null.

## Constructors

---

### HVideoDevice()

```
protected HVideoDevice()
```

It is not intended that applications should directly construct `HVideoDevice` objects.

Creates an `HVideoDevice` object. See the class description for details of constructor parameters and default values.

## Methods

---

### getBestConfiguration(HVideoConfigTemplate)

```
public HVideoConfiguration getBestConfiguration(HVideoConfigTemplate hvct)
```

Returns the "best" configuration possible that passes the criteria defined in the `HVideoConfigTemplate` or null.

Equally best in this sense means that the configurations satisfy an equal number of preferences with priorities `PREFERRED` and `PREFERRED_NOT` and all preferences with priorities `REQUIRED` and `REQUIRED_NOT`.

Configurations are chosen according to the following algorithm, based on the priority as supplied to `setPreference(int, int)`. Configurations must:

1. satisfy ALL the preferences whose priority was `REQUIRED`
2. satisfy NONE of the preferences whose priority was `REQUIRED_NOT`
3. satisfy as many as possible of the preferences whose priority was `PREFERRED`.
4. satisfy as few as possible of the preferences whose priority was `PREFERRED_NOT`.

Preferences whose priority was `DONT_CARE` are ignored.

This method returns null if no configuration exists that satisfies all `REQUIRED` and `REQUIRED_NOT` priorities.

#### Parameters:

`hvct` - an `HVideoConfigTemplate` object used to obtain a valid `HVideoConfiguration`. If this parameter is null the default configuration for the platform shall be returned.

#### Returns:

an `HVideoConfiguration` that passes the criteria defined in the specified `HVideoConfigTemplate` or null if no `HVideoConfiguration` passes the criteria.

---

### getBestConfiguration(HVideoConfigTemplate[])

```
public HVideoConfiguration getBestConfiguration(HVideoConfigTemplate[] hvcta)
```

Returns the "best" configuration possible that passes the criteria defined in one of the `HVideoConfigTemplate` objects within the specified array or null. The `HVideoTemplate` objects should be considered for matching in priority order from 0 to (`hvcta.length - 1`).

Equally best in this sense means that the configurations satisfy an equal number of preferences with priorities `PREFERRED` and `PREFERRED_NOT` and all preferences with priorities `REQUIRED` and `REQUIRED_NOT`.

Configurations are chosen according to the following algorithm, based on the priority as supplied to `setPreference(int, int)`. Configurations must:

1. satisfy ALL the preferences whose priority was `REQUIRED`
2. satisfy NONE of the preferences whose priority was `REQUIRED_NOT`
3. satisfy as many as possible of the preferences whose priority was `PREFERRED`.
4. satisfy as few as possible of the preferences whose priority was `PREFERRED_NOT`.

Preferences whose priority was `DONT_CARE` are ignored.

This method returns null if no configuration exists that satisfies all `REQUIRED` and `REQUIRED_NOT` priorities.

**Parameters:**

`hvcta` - the `HVideoConfigTemplate` array used to obtain a valid `HVideoConfiguration`.

**Returns:**

an `HVideoConfiguration` that passes the criteria defined in one of the `HVideoConfigTemplate` objects within the specified array

### **getConfigurations()**

```
public HVideoConfiguration[] getConfigurations()
```

Returns all of the `HVideoConfiguration` objects associated with this `HVideoDevice`.

**Returns:**

an array of `HVideoConfiguration` objects

**See Also:**

`HVideoConfiguration`

### **getCurrentConfiguration()**

```
public HVideoConfiguration getCurrentConfiguration()
```

Returns the current `HVideoConfiguration` for this `HVideoDevice`.

**Returns:**

the current `HVideoConfiguration` for this `HVideoDevice`.

**See Also:**

`HVideoConfiguration`

### **getDefaultConfiguration()**

```
public HVideoConfiguration getDefaultConfiguration()
```

Returns the default `HVideoConfiguration` associated with this `HVideoDevice`. This (single) default configuration should correspond to some well-behaved settings for the device, such as, a minimal configuration or factory preset settings.

**Returns:**

the default `HVideoConfiguration` of this `HVideoDevice`.

**See Also:**

[HVideoConfiguration](#)

---

**getVideoController()**

```
public java.lang.Object getVideoController()
```

Obtain a reference to the object which controls the presentation of the video. Null is returned if no video is being presented. In systems based on JMF, this would be the `javax.media.Player` instance which owns the resource.

**Returns:**

the object which controls the presentation of the video

**Throws:**

`SecurityException` - if the application does not have sufficient rights to get the `VideoPlayer` object.

`HPermissionDeniedException` - (`HPermissionDeniedException`) if the application does not currently have the right to get the `VideoPlayer` object.

---

**getVideoSource()**

```
public java.lang.Object getVideoSource()
```

Obtain a reference to the source of the video being presented by this device at this moment. The precise class to be returned must be specified outside the HAVi user- interface specification. Null is returned if no video is being presented.

**Returns:**

a reference to the source of the video

**Throws:**

`SecurityException` - if the application does not have sufficient rights to get the `VideoSource` object.

`HPermissionDeniedException` - (`HPermissionDeniedException`) if the application does not currently have the right to get the `VideoSource` object.

---

**setVideoConfiguration(HVideoConfiguration)**

```
public boolean setVideoConfiguration(HVideoConfiguration hvc)
```

Set the video configuration for the device.

An application is only allowed to call this method after it reserved the device explicitly and subject to the security policy of the platform. Subject to this, the following rules determine whether this method can succeed.

- If the application selects a configuration that is not conflicting with the configurations of all the other devices on the same `HScreen` then that configuration is selected.
- If an application tries to select a configuration that conflicts with a configuration of another device on this `HScreen` which this application cannot control due to platform security policy then this method shall fail with a `SecurityException`.
- If an application tries to select a configuration that conflicts with a configuration of another device on this `HScreen` which this application cannot control due to another application owning the right to control that device and the platform not giving that right to this application then this method shall fail with an `HPermissionDeniedException`.
- If an application tries to select a configuration that conflicts with a configuration of another device which this application can control due to either this or no application having reserved the device, then the configuration of the other device is automatically changed. There is no change in the resource ownership of the

other device.

Applications can prevent or limit changes to configurations of other, not intended, devices by using constants `ZERO_GRAPHICS_IMPACT` and `ZERO_VIDEO_IMPACT` in their configuration templates. The first one will cause no changes to already running graphical applications. This means that no changes may be applied to the graphics device. Similarly the second constant will result in no changes to the video device.

Any modifications made to the configurations of other devices shall be reflected by the API for the devices concerned. The `getCurrentConfiguration()` method for those devices shall return the new configuration.

On successful change to the specified configuration, the device shall fire one or more `HScreenConfigurationEvent` for all listeners that are currently registered (if the criteria for receiving such an event has been satisfied). If the new configuration differs by more than one characteristic from the previous configuration then the device may fire one or more `HScreenConfigurationEvent`. This behavior is implementation specific.

If an attempt is made to set the identical configuration to that which is current, then no such event(s) shall be fired.

**Parameters:**

`hvc` - the `HVideoConfiguration` to which this device should be set.

**Returns:**

A boolean indicating whether the configuration could be applied successfully. If the configuration could not be applied successfully, the configuration after this method may not match the configuration of the device prior to this method being called --- applications should take steps to determine whether a partial change of settings has been made.

**Throws:**

`SecurityException` - if the application does not have sufficient rights to set the configuration for this device.

{ [@link - org.havi.ui.HPermissionDeniedException](#) `HPermissionDeniedException`} if the application does not currently have the right to set the configuration for this device.

{ [@link - org.havi.ui.HConfigurationException](#) `HConfigurationException`} if the specified configuration is not valid for this device, or if it conflicts with other devices whose configuration(s) cannot be changed by this application.

`HConfigurationException`, `HPermissionDeniedException`



# org.havi.ui

## HVisible

### Syntax

```
public class HVisible extends HComponent implements HState
```

```
java.lang.Object
|
+--java.awt.Component
|
+--HComponent
|
+--org.havi.ui.HVisible
```

### Direct Known Subclasses:

```
HListGroup, HSinglelineEntry, HStaticAnimation, HStaticIcon, HStaticRange,
HStaticText
```

### All Implemented Interfaces:

```
HMatteLayer, HState, java.awt.image.ImageObserver, java.awt.MenuContainer,
java.io.Serializable, TestOpacity
```

### Description

The `HVisible` class is the base class for all non-interactive components.

If a layout manager is associated with the Container into which a `HVisible` component is placed, the size and location of the component will be controlled by the layout manager.

`HVisible` provides the following features for the use of subclasses:

- support for interaction states.
- a mechanism to associate the component with a pluggable `HLook` class to which all drawing is delegated.
- support for state-related content which is drawn by the associated `HLook`.
- support for scalable and alignable content.
- support for private look data.
- support for preferred sizes when used with a layout manager.
- control over the background painting behavior.
- a pluggable text layout management mechanism.

Some of these features are discussed in more detail below.

#### Interaction State

Interaction states for `HVisible` components are defined in the `HState` interface. The only interaction states valid for a plain (un-subclassed) `HVisible` are:

- `NORMAL_STATE` - indicating that the component is in its normal interaction state.
- `DISABLED_STATE` - indicating that the component is disabled. While this has no effect on user interaction with a plain `HVisible` (since there is no user interaction) it will affect the visual appearance.

Subclasses of `HVisible` may use other states. However, `setInteractionState(int)` is the only means by which state can be changed and will throw an `IllegalArgumentException` if the state is not valid for a particular component type. Subclasses of `HVisible` must not manipulate state in other ways.

The `getInteractionState()` method is provided to allow any class to read the current state of a component.

#### State-based content

Specific content may be set for any interaction state defined in `HState`, irrespective of whether the subclass of `HVisible` can ever be in that state.

Note that content is set on the various `STATE` constants defined in `HState`, and *not* on the `STATE_BIT` constants. A `java.lang.IllegalArgumentException` will be thrown by any method which takes a state as a parameter if a `STATE_BIT` rather than a `STATE` is passed.

By default it is possible to set content for any of eight discrete states. However, if no content has been set for a particular state, the associated `HLook` should use content from the nearest matching state as defined below:

State	Content if missing	Example style
<code>NORMAL_STATE</code>	none	no special style
<code>FOCUSED_STATE</code>	<code>NORMAL_STATE</code>	highlighted
<code>ACTIONED_STATE</code>	<code>FOCUSED_STATE</code>	pushed in
<code>ACTIONED_FOCUSED_STATE</code>	<code>FOCUSED_STATE</code>	highlighted & pushed in
<code>DISABLED_STATE</code>	<code>NORMAL_STATE</code>	grayed out
<code>DISABLED_FOCUSED_STATE</code>	<code>DISABLED_STATE</code>	grayed out & highlighted
<code>DISABLED_ACTIONED_STATE</code>	<code>ACTIONED_STATE</code>	grayed out & pushed in
<code>DISABLED_ACTIONED_FOCUSED_STATE</code>	<code>DISABLED_STATE</code>	grayed out & highlighted & pushed

It is recommended that `HLook` implementations be capable of drawing components in styles that allow the interaction state to be visually distinguished even when no content is set, or the content is the same for several states.

Most content is stored by reference, but text content is copied as `java.lang.String` objects are immutable.

Content for any component may be changed "on the fly" using the `setContent` methods. However, some components (e.g. `HStaticAnimation` and `HAnimation`) may be asynchronously referencing their content (i.e. through a separate implementation-specific animation thread). Therefore the following restrictions apply to the `setAnimateContent(Image[], int)` method when the instance of `HVisible` on which it is invoked implements the `HAnimateEffect` interface:

- The method must be synchronized with any implementation-specific animation thread such that content cannot be changed while a different thread is using it.
- If the animation was running the method should stop the animation in a synchronized manner before changing content.
- The method should reset the animation to a starting position defined by the current play mode. The repeat count of the animation should be reset to 0.
- If the animation was running the method should start the animation.

Alternative platform-specific animation schemes which do not use `HAnimateEffect` are outside the scope of this document. However, a similar strategy should be employed to avoid synchronization problems.

#### Content Scaling and Alignment

Where state-based content is used, it is an implementation option to support scaling of the content to fit the `HVisible`. Some looks also support alignment of content. See the class description of `HLook` for

details of which looks support scaling and alignment. See the fields description of this class for constants which can be used to specify scaling and alignment modes.

The following methods are used to get and set alignment of content:

- `setHorizontalAlignment(int)`
- `setVerticalAlignment(int)`
- `getHorizontalAlignment()`
- `getVerticalAlignment()`

The following methods are used to get and set the scaling of content. Scaling support is optional, however all implementations must support the `RESIZE_NONE` scaling mode. Platforms are *not* required to support scaling of textual content by default.

- `setResizeMode(int)`
- `getResizeMode()`
- Repaint Hints

An associated `HLook` provides a mechanism of efficiently repainting an `HVisible`, based on a hint which provides data about the change which triggered the redrawing. The `widgetChanged(HVisible, HChangeData[])` method should be called in preference to `repaint` whenever the `HVisible` requires a redraw to change its appearance.

The `widgetChanged(HVisible, HChangeData[])` method requires an array of one or more `HChangeData` objects, which contain a hint, and associated data. All keys are `java.lang.Integer` objects, with values defined as constants in this class. The following hints are defined:

Hint (key)	Data (value)	Data Interpretation
<code>TEXT_CONTENT_CHANGE</code>	<code>Object[9]</code>	[Integer <code>changedState</code> , String <code>oldNORMAL_STATEtext</code> , String <code>oldFOCUSED_STATEtext</code> , String <code>oldACTIONED_STATEtext</code> , String <code>oldACTIONED_FOCUSED_STATEtext</code> , String <code>oldDISABLED_STATEtext</code> , String <code>oldDISABLED_FOCUSED_STATEtext</code> , String <code>oldDISABLED_ACTIONED_STATEtext</code> , String <code>oldDISABLED_ACTIONED_FOCUSED_STATEtext</code> ]
<code>GRAPHIC_CONTENT_CHANGE</code>	<code>Object[9]</code>	[Integer <code>changedState</code> , Image <code>oldNORMAL_STATEimage</code> , Image <code>oldFOCUSED_STATEimage</code> , Image <code>oldACTIONED_STATEimage</code> , Image <code>oldACTIONED_FOCUSED_STATEimage</code> , Image <code>oldDISABLED_STATEimage</code> , Image <code>oldDISABLED_FOCUSED_STATEimage</code> , Image <code>oldDISABLED_ACTIONED_STATEimage</code> , Image <code>oldDISABLED_ACTIONED_FOCUSED_STATEimage</code> ]
<code>ANIMATE_CONTENT_CHANGE</code>	<code>Object[9]</code>	[Integer <code>changedState</code> , Image[] <code>oldNORMAL_STATEanimation</code> , Image[] <code>oldFOCUSED_STATEanimation</code> , Image[] <code>oldACTIONED_STATEanimation</code> , Image[] <code>oldACTIONED_FOCUSED_STATEanimation</code> , Image[] <code>oldDISABLED_STATEanimation</code> , Image[] <code>oldDISABLED_FOCUSED_STATEanimation</code> , Image[] <code>oldDISABLED_ACTIONED_STATEanimation</code> , Image[] <code>oldDISABLED_ACTIONED_FOCUSED_STATEanimation</code> ]

CONTENT_CHANGE	Object[9]	[Integer changedState, Object oldNORMAL_STATEcontent, Object oldFOCUSED_STATEcontent, Object oldACTIONED_STATEcontent, Object oldACTIONED_FOCUSED_STATEcontent, Object oldDISABLED_STATEcontent, Object oldDISABLED_FOCUSED_STATEcontent, Object oldDISABLED_ACTIONED_STATEcontent, Object oldDISABLED_ACTIONED_FOCUSED_STATEcontent]
STATE_CHANGE	Integer	oldState
CARET_POSITION_CHANGE	Integer	oldPosition
ECHO_CHAR_CHANGE	Character	oldEcho
EDIT_MODE_CHANGE	Boolean	oldEditMode
MIN_MAX_CHANGE	Integer[2]	[oldMin, oldMax]
THUMB_OFFSETS_CHANGE	Integer[2]	[oldMin, oldMax]
ORIENTATION_CHANGE	Integer	oldOrientation
TEXT_VALUE_CHANGE	Integer	oldValue
ITEM_VALUE_CHANGE	Integer	oldValue
ADJUSTMENT_VALUE_CHANGE	Integer	oldIndex
LIST_CONTENT_CHANGE	HListElement[]	oldContent
LIST_ICONSIZING_CHANGE	Dimension	oldSize
LIST_LABELSIZING_CHANGE	Dimension	oldSize
LIST_MULTISELECTION_CHANGE	Boolean	oldSelection
LIST_SCROLLPOSITION_CHANGE	Integer	oldScrollPosition
UNKNOWN_CHANGE	Integer	UNKNOWN_CHANGE

Note that implementations of `HLook` may not actually implement more efficient drawing code for a given hint. In particular, simply repainting the entire `HVisible` is a valid implementation option.

The following code for the `setThumbOffsets(int, int)` method demonstrates how hint objects are used. Note that the values passed in the hint are always the *old* values, since the `HLook` can always retrieve the new values as needed from the `HVisible`. Variables starting with "my" are member variables of the class.

```

public void setThumbOffsets(int min, int max)
{
    int oldValue = myValue;
    HChangeData[] array = new HChangeData[1];
    array[0] = null;
    // valid argument checking elided
    // create hint object with the OLD values
    HChangeData d = new HChangeData(HVisible.MIN_MAX_CHANGE,
    new Integer[] { new Integer(myMin),
    new Integer(myMax) });
    // update the values
    myMin = min;
    myMax = max;

    // check bounds
    myValue = Math.min(myMin, Math.max(myMax, myValue));

    // create value hint if needed
    if (myValue != oldValue)
        array[0] = d;

    // tell the look to repaint as needed
    myLook.widgetChanged(this, array);
}

```

Private HLook data

Private data for the use of an associated `HLook` to optimize the redraw of the component may be set on any `HVisible`. Examples of such data are:

- cached bitmap representations of complex drawing operations
- scrolling metrics and positions
- cached layout data

Data is set using the `setLookData(Object, Object)` method, and retrieved with the `getLookData(Object)` method.

To keep the on-screen representation of an `HVisible` synchronized with its state, any class may call `setLookData(Object, Object)` with a parameter of `null` to invalidate any cached data. Instances of `HLook` must be able to regenerate any data they store on `HVisible` classes at any time. This mechanism is intended for implementation optimization only and shall not be used for storing content.

Use of this mechanism is an implementation option. If this mechanism is not used by an implementation, `getLookData(Object)` shall always return `null` and `setLookData(Object, Object)` shall do nothing. Interoperable systems shall not assume that this mechanism is implemented.

#### Preferred Sizes

The `setDefaultSize(Dimension)` and `getDefaultSize()` methods provide support for a user-defined default preferred size to be passed to a layout manager. This value is returned to the layout manager through the `getPreferredSize(HVisible)` method and if set overrides any look-specific value based on content calculated by the `HLook`.

Note that the interpretation of the size passed to `setDefaultSize(Dimension)` is *not* the overall size of the component, but the area available to the look to render the component into excluding any look-specific borders. Therefore when a layout manager is in use the *actual* size of the component is likely to be larger than this default size. See the descriptions of `getMinimumSize(HVisible)`, `getPreferredSize(HVisible)` and `getMaximumSize(HVisible)` in `HLook` for details of the exact algorithm used.

Note that constructors of `HVisible` and subclasses which accept `width` and `height` parameters shall set the default size to `[width, height]` as if these parameters were passed to `setDefaultSize(Dimension)`. If a layout manager is in use the actual size of the component will probably be larger than this size after layout due to the `HLook` adding borders. However, if no layout manager is used the `width` and `height` parameters are simply used to set the *actual* size of the component.

#### Background Painting

Normally the associated `HLook` does not paint the background of the `HVisible`, allowing for non-rectangular components and text overlaying bitmaps. However, `HVisible` provides for components which require their background to be painted through the `setBackgroundMode` method. Note that if the mode is set to `BACKGROUND_FILL` the return value of the `isOpaque()` method *may* be true, depending on whether the current background color of the `HVisible` is opaque. If the background mode is set to `NO_BACKGROUND_FILL` the `isOpaque()` method *must* return false.

#### Event Handling

While implementations of `HVisible` may enable certain `java.awt.AWTEvents`, applications should assume that an `HVisible` class does not generate or respond to any `java.awt.AWTEvents`. If this behavior is required the standard AWT mechanisms (i.e. `processEvent` and similar functions) may be used to handle events on `HVisible` or subclasses. However, it is strongly recommended that component implementors use the HAVi events defined in the `org.havi.ui.event` package where possible.

For example, a component wishing to respond to user action should normally subclass `HComponent`, implement the `HActionInputPreferred` interface and handle `HActionEvent` events in the `processHActionEvent(HActionEvent)` method, instead of overriding `java.awt.Component#processEvent` or similar. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

#### Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
width	width of this component in pixels (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>
height	height of this component in pixels (subject to layout management).	---	<code>java.awt.Component#setBounds</code>	<code>java.awt.Component#getBounds</code>

#### Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte ( <code>HMatte</code> ).	none (i.e. <code>getMatte()</code> returns null)	<code>setMatte(HMatte)</code>	<code>getMatte()</code>
The text layout manager responsible for text formatting.	An <code>HDefaultTextLayoutManager</code> object.	<code>setTextLayoutManager(HTextLayoutManager)</code>	<code>getTextLayoutManager()</code>
The background painting mode	<code>NO_BACKGROUND_FILL</code>	<code>setBackgroundMode(int)</code>	<code>getBackgroundMode()</code>

The default preferred size	not set (i.e. null) unless specified by width and height parameters	<code>setDefaultSize(Dimension)</code>	<code>getDefaultSize()</code>
The horizontal content alignment	<code>HALIGN_CENTER</code>	<code>setHorizontalAlignment(int)</code>	<code>getHorizontalAlignment()</code>
The vertical content alignment	<code>VALIGN_CENTER</code>	<code>setVerticalAlignment(int)</code>	<code>getVerticalAlignment()</code>
The content scaling mode	<code>RESIZE_NONE</code>	<code>setResizeMode(int)</code>	<code>getResizeMode()</code>

## Fields

---

### ADJUSTMENT\_VALUE\_CHANGE

```
public static final int ADJUSTMENT_VALUE_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that the value of an `HAdjustmentValue` component has changed. The value for this hint is a `java.lang.Integer` which contains the old index.

---

### ANIMATE\_CONTENT\_CHANGE

```
public static final int ANIMATE_CONTENT_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that the animated content has changed. The value for this hint is an array, `java.lang.Object[ 9 ]`, which contains the state for which the content changed (a `java.lang.Integer`) and the old content (a `java.awt.Image[ ]`) for all 8 states.

---

### BACKGROUND\_FILL

```
public static final int BACKGROUND_FILL
```

A constant for use with the `setBackgroundMode(int)` method of `HVisible` which indicates that an associated `HLook` should fill the bounding rectangle of the `HVisible` with its current background color before drawing any content. Any previous content will be erased during the repainting of the `HVisible`.

---

### CARET\_POSITION\_CHANGE

```
public static final int CARET_POSITION_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that the caret position has changed. The value for this hint is a `java.lang.Integer` which has the value of the old caret position.

---

## CONTENT\_CHANGE

```
public static final int CONTENT_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that the miscellaneous content has changed. The value for this hint is an array, `java.lang.Object[ 9]`, which contains the state for which the content changed (a `java.lang.Integer`) and the old content (a `java.lang.Object`) for all 8 states.

---

## ECHO\_CHAR\_CHANGE

```
public static final int ECHO_CHAR_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that the echo character has changed. The value for this hint is a `java.lang.Character` which has the value of the old echo character.

---

## EDIT\_MODE\_CHANGE

```
public static final int EDIT_MODE_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that the editing mode has changed. The value for this hint is a `java.lang.Boolean` which has the value of the old edit mode.

---

## FIRST\_CHANGE

```
public static final int FIRST_CHANGE
```

A constant which indicates the first change value for use with the hinting mechanism.

---

## GRAPHIC\_CONTENT\_CHANGE

```
public static final int GRAPHIC_CONTENT_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that the graphical content has changed. The value for this hint is an array, `java.lang.Object[ 9]`, which contains the state for which the content changed (a `java.lang.Integer`) and the old content (a `java.awt.Image`) for all 8 states.

---

## HALIGN\_CENTER

```
public static final int HALIGN_CENTER
```

A constant for use with the `setHorizontalAlignment(int)` method of `HVisible` which indicates that content should be centered horizontally.

---

## HALIGN\_JUSTIFY

```
public static final int HALIGN_JUSTIFY
```

A constant for use with the `setHorizontalAlignment(int)` method of `HVisible` which indicates that content should be fully justified (horizontally).



---

**HALIGN\_LEFT**

```
public static final int HALIGN_LEFT
```

A constant for use with the `setHorizontalAlignment(int)` method of `HVisible` which indicates that content should be left aligned.

---

**HALIGN\_RIGHT**

```
public static final int HALIGN_RIGHT
```

A constant for use with the `setHorizontalAlignment(int)` method of `HVisible` which indicates that content should be right aligned.

---

**ITEM\_VALUE\_CHANGE**

```
public static final int ITEM_VALUE_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that the value of an `HItemValue` component has changed. The value for this hint is a `java.lang.Integer` which contains the old value.

---

**LAST\_CHANGE**

```
public static final int LAST_CHANGE
```

A constant which indicates the last defined value for use with the hinting mechanism.

---

**LIST\_CONTENT\_CHANGE**

```
public static final int LIST_CONTENT_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that the content of an `HListGroup` component has changed. The value for this hint is a `java.lang.Integer` which contains the old content.

---

**LIST\_ICONSIZING\_CHANGE**

```
public static final int LIST_ICONSIZING_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that the iconsize of an `HListGroup` component has changed. The value for this hint is a `java.lang.Integer` which contains the old size.

---

**LIST\_LABELSIZING\_CHANGE**

```
public static final int LIST_LABELSIZING_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that the labelsize of an `HListGroup` component has changed. The value for this hint is a `java.lang.Integer` which contains the old size.

---

**LIST\_MULTISELECTION\_CHANGE**

```
public static final int LIST_MULTISELECTION_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that the multiselection setting of an `HListGroup` component has changed. The value for this hint is a `java.lang.Integer` which contains the old setting.

---

**LIST\_SCROLLPOSITION\_CHANGE**

```
public static final int LIST_SCROLLPOSITION_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that the scrollposition of an `HListGroup` component has changed. The value for this hint is a `java.lang.Integer` which contains the old position.

---

**MIN\_MAX\_CHANGE**

```
public static final int MIN_MAX_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that the range of an `HAdjustmentValue` component has changed. The value for this hint is an array, `java.lang.Object[ 2]`, which contains the old minimum and maximum extents of the range as `java.lang.Integer`.

---

**NO\_BACKGROUND\_FILL**

```
public static final int NO_BACKGROUND_FILL
```

A constant for use with the `setBackgroundMode(int)` method of `HVisible` which indicates that an associated `HLook` should not fill the bounding rectangle of the `HVisible` with its current background color before drawing any content. Therefore any previous content will NOT necessarily be erased during the repainting of the `HVisible`.

---

**NO\_DEFAULT\_HEIGHT**

```
public static final int NO_DEFAULT_HEIGHT
```

A constant for use with the `setDefaultSize(Dimension)` and `getDefaultSize()` methods of `HVisible` which indicates that no default height is desired for the `HVisible`.

---

**NO\_DEFAULT\_SIZE**

```
public static final java.awt.Dimension NO_DEFAULT_SIZE
```

A constant for use with the `setDefaultSize(Dimension)` and `getDefaultSize()` methods of `HVisible` which indicates that no default size is desired for the `HVisible`.

---

**NO\_DEFAULT\_WIDTH**

```
public static final int NO_DEFAULT_WIDTH
```

A constant for use with the `setDefaultSize(Dimension)` and `getDefaultSize()` methods of `HVisible` which indicates that no default width is desired for the `HVisible`.

---

**ORIENTATION\_CHANGE**

```
public static final int ORIENTATION_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that the orientation of an `HOrientable` component has changed. The value for this hint is a `java.lang.Integer` which has the value of the old orientation.

---

**RESIZE\_ARBITRARY**

```
public static final int RESIZE_ARBITRARY
```

A constant for use with the `setResizeMode(int)` method of `HVisible` which indicates that content should be scaled to fit the component. Aspect ratios of the content need not be preserved.

---

**RESIZE\_NONE**

```
public static final int RESIZE_NONE
```

A constant for use with the `setResizeMode(int)` method of `HVisible` which indicates that content should not be scaled to fit the component.

---

**RESIZE\_PRESERVE\_ASPECT**

```
public static final int RESIZE_PRESERVE_ASPECT
```

A constant for use with the `setResizeMode(int)` method of `HVisible` which indicates that content should be scaled to fit the component while preserving the aspect ratio of the content. Areas of the component that are not filled by the content will be look dependent.

---

**STATE\_CHANGE**

```
public static final int STATE_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that the interaction state has changed. The value for this hint is a `java.lang.Integer` which has the value of the old state.

---

**TEXT\_CONTENT\_CHANGE**

```
public static final int TEXT_CONTENT_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that the text content has changed. The value for this hint is an array, `java.lang.Object[ 9]`, which contains the state for which the content changed (a `java.lang.Integer`) and the old content (a `java.lang.String`) for all 8 states.

---

**TEXT\_VALUE\_CHANGE**

```
public static final int TEXT_VALUE_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that the value of an `HTextValue` component has changed. The value for this hint is a `java.lang.Integer` which contains the old value.

---

## THUMB\_OFFSETS\_CHANGE

```
public static final int THUMB_OFFSETS_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that the thumb offsets of an `HAdjustmentValue` component have changed. The value for this hint is an array, `java.lang.Object[ 2]`, which contains the old minimum and maximum thumb offsets as `java.lang.Integer`.

---

## UNKNOWN\_CHANGE

```
public static final int UNKNOWN_CHANGE
```

A constant for use with the hinting mechanism (see the `widgetChanged(HVisible, HChangeData[])` method in `HLook`). This hint indicates that some unspecified change has occurred. The value for this hint is a `java.lang.Integer` which also has the value `UNKNOWN_CHANGE`.

---

## VALIGN\_BOTTOM

```
public static final int VALIGN_BOTTOM
```

A constant for use with the `setVerticalAlignment(int)` method of `HVisible` which indicates that content should be vertically aligned to the bottom of the component.

---

## VALIGN\_CENTER

```
public static final int VALIGN_CENTER
```

A constant for use with the `setVerticalAlignment(int)` method of `HVisible` which indicates that content should be centered vertically.

---

## VALIGN\_JUSTIFY

```
public static final int VALIGN_JUSTIFY
```

A constant for use with the `setVerticalAlignment(int)` method of `HVisible` which indicates that content should be fully justified (vertically).

---

## VALIGN\_TOP

```
public static final int VALIGN_TOP
```

A constant for use with the `setVerticalAlignment(int)` method of `HVisible` which indicates that content should be vertically aligned to the top of the component.

## Constructors

---

### HVisible()

```
public HVisible()
```

Creates an `HVisible` component with no `HLook`. See the class description for details of constructor parameters and default values.

---

## HVisible(HLook)

```
public HVisible(HLook hlook)
```

Creates an `HVisible` component with the given `HLook`. See the class description for details of constructor parameters and default values.

**Parameters:**

`hlook` - The `HLook` associated with the `HVisible` component.

---

## HVisible(HLook, int, int, int, int)

```
public HVisible(HLook hlook, int x, int y, int width, int height)
```

Creates an `HVisible` component with the given `HLook` and the specified location and size. See the class description for details of constructor parameters and default values.

**Parameters:**

`hlook` - The `HLook` associated with the `HVisible` component.

`x` - the x-coordinate of the `HVisible` component within its Container.

`y` - the y-coordinate of the `HVisible` component within its Container.

`width` - the width of the `HVisible` component in pixels.

`height` - the height of the `HVisible` component in pixels.

## Methods

---

### getAnimateContent(int)

```
public java.awt.Image[] getAnimateContent(int state)
```

Gets the animate content for this component.

**Parameters:**

`state` - The state for which content is to be retrieved. Note that content is set on the `XXX_STATE` constants defined in `HState`, and *not* on the `XXX_STATE_BIT` constants. A `java.lang.IllegalArgumentException` will be thrown if a `STATE_BIT` rather than a `STATE` is passed.

**Returns:**

The animate content associated with the specified state. If no animate content has been set for the specified state, then `null` is returned.

---

### getBackgroundMode()

```
public int getBackgroundMode()
```

Get the background mode of this `HVisible`. The return value specifies how the look should draw the background (i.e. a rectangle filling the bounds of the `HVisible`).

**Returns:**

one of `NO_BACKGROUND_FILL` or `BACKGROUND_FILL`.

---

### getContent(int)

```
public java.lang.Object getContent(int state)
```

Gets the content for this component.

**Parameters:**

`state` - The state for which content is to be retrieved. Note that content is set on the `XXX_STATE` constants defined in `HState`, and *not* on the `XXX_STATE_BIT` constants. A `java.lang.IllegalArgumentException` will be thrown if a `STATE_BIT` rather than a `STATE` is passed.

**Returns:**

The content associated with the specified state. If no content has been set for the specified state, then `null` is returned.

---

**getDefaultSize()**

```
public java.awt.Dimension getDefaultSize()
```

Returns the default preferred size to be used for this component when a layout manager is in use. If no default size has been set using the `setDefaultSize(Dimension)` method `NO_DEFAULT_SIZE` is returned.

If the parent Container into which the `HVisible` is placed has no layout manager the default preferred size has no effect.

The default size of a component is to be interpreted as the area in which the component can be rendered, *excluding* look-specific borders.

**Returns:**

the default preferred size to be used for this component when a layout manager is in use.

---

**getGraphicContent(int)**

```
public java.awt.Image getGraphicContent(int state)
```

Gets the graphic content for this component.

**Parameters:**

`state` - The state for which content is to be retrieved. Note that content is set on the `XXX_STATE` constants defined in `HState`, and *not* on the `XXX_STATE_BIT` constants. A `java.lang.IllegalArgumentException` will be thrown if a `STATE_BIT` rather than a `STATE` is passed.

**Returns:**

The graphical content associated with the specified state. If no graphical content has been set for the specified state, then `null` is returned.

---

**getHorizontalAlignment()**

```
public int getHorizontalAlignment()
```

Get the horizontal alignment of any state-based content rendered by an associated `HLook`. If content is not used in the rendering of this `HVisible` the value returned shall be valid, but has no affect on the rendered representation.

**Returns:**

the current horizontal alignment mode, one of `HALIGN_LEFT`, `HALIGN_CENTER`, `HALIGN_RIGHT` or `HALIGN_JUSTIFY`.

---

**getInteractionState()**

```
public int getInteractionState()
```

Return the interaction state the component is currently in.

**Returns:**

the interaction state the component is currently in.

**See Also:**

[HState](#)

---

**getLook()**

```
public HLook getLook()
```

Gets the [HLook](#) for this component.

**Returns:**

the [HLook](#) that is being used by this component - if no [HLook](#) has been set, then returns null.

---

**getLookData(Object)**

```
public java.lang.Object getLookData(java.lang.Object key)
```

Retrieve a look-specific private data object. Instances of [HLook](#) may use this method to retrieve private data (e.g. layout hints, cached images etc.) from the [HVisible](#). Use of this mechanism is an implementation option. If this mechanism is not used by an implementation, or no data has been set for the specified key this method returns null.

**Parameters:**

*key* - an object which uniquely identifies the type of look for which the private data is to be retrieved. Keys need not be unique across different instances of the same look class.

**Returns:**

a private data object as previously set using [setLookData\(Object, Object\)](#) , or null.

**See Also:**

[HLook](#), [setLookData\(Object, Object\)](#)

---

**getMaximumSize()**

```
public java.awt.Dimension getMaximumSize()
```

Gets the maximum size of the [HVisible](#). The [getMaximumSize](#) method of the [HLook](#) that is associated with this [HVisible](#) will be called to calculate the dimensions.

**Overrides:**

[java.awt.Component.getMaximumSize\(\)](#) in class [java.awt.Component](#)

**Returns:**

A dimension object indicating this [HVisible](#) 's maximum size - if no [HLook](#) has been associated with the [HVisible](#) , then the current [HVisible](#) dimensions as determined with [getSize](#) will be returned.

**See Also:**

[getMaximumSize\(HVisible\)](#)

---

**getMinimumSize()**

```
public java.awt.Dimension getMinimumSize()
```

Gets the minimum size of the [HVisible](#). The [getMinimumSize](#) method of the [HLook](#) that is associated with this [HVisible](#) will be called to calculate the dimensions.

**Overrides:**

[java.awt.Component.getMinimumSize\(\)](#) in class [java.awt.Component](#)

---

**Returns:**

A dimension object indicating this `HVisible` 's minimum size - if no `HLook` has been associated with the `HVisible` , then the current `HVisible` dimensions as determined with `getSize` will be returned.

**See Also:**

`getMinimumSize(HVisible)`

---

**getPreferredSize()**

```
public java.awt.Dimension getPreferredSize()
```

Gets the preferred size of the `HVisible` . The `getPreferredSize` method of the `HLook` that is associated with this `HVisible` will be called to calculate the dimensions.

**Overrides:**

`java.awt.Component.getPreferredSize()` in class `java.awt.Component`

**Returns:**

A dimension object indicating this `HVisible` 's preferred size - if no `HLook` has been associated with the `HVisible` , then the current `HVisible` dimensions as determined with `getSize` will be returned.

**See Also:**

`getPreferredSize(HVisible)`

---

**getResizeMode()**

```
public int getResizeMode()
```

Get the scaling mode for scaling any state-based content rendered by an associated `HLook` . If content is not used in the rendering of this `HVisible` the value returned shall be valid, but has no affect on the rendered representation.

**Returns:**

the current scaling mode, one of `RESIZE_NONE` , `RESIZE_PRESERVE_ASPECT` or `RESIZE_ARBITRARY`

---

**getTextContent(int)**

```
public java.lang.String getTextContent(int state)
```

Gets the text content for this component.

**Parameters:**

`state` - The state for which content is to be retrieved. Note that content is set on the `XXX_STATE` constants defined in `HState` , and *not* on the `XXX_STATE_BIT` constants. A `java.lang.IllegalArgumentException` will be thrown if a `STATE_BIT` rather than a `STATE` is passed.

**Returns:**

The text content associated with the specified state. If no text content has been set for the specified state, then `null` is returned.

---

**getTextLayoutManager()**

```
public HTextLayoutManager getTextLayoutManager()
```

Gets the text layout manager that is being used to layout this text.

**Returns:**



The `HTextLayoutManager` that is being used by this component.

---

### **getVerticalAlignment()**

```
public int getVerticalAlignment()
```

Get the vertical alignment of any state-based content rendered by an associated `HLook`. If content is not used in the rendering of this `HVisible` the value returned shall be valid, but has no affect on the rendered representation.

**Returns:**

the current vertical alignment mode, one of `VALIGN_TOP`, `VALIGN_CENTER`, `VALIGN_BOTTOM` or `VALIGN_JUSTIFY`.

---

### **isFocusTraversable()**

```
public boolean isFocusTraversable()
```

By default an `HVisible` component is not focus-traversable.

**Overrides:**

`java.awt.Component.isFocusTraversable()` in class `java.awt.Component`

**Returns:**

false

**See Also:**

`java.awt.Component.isFocusTraversable()`

---

### **isOpaque()**

```
public boolean isOpaque()
```

Returns true if the entire `HVisible` area, as given by the `java.awt.Component#getBounds` method, is fully opaque, i.e. its `HLook` guarantees that all pixels are painted in an opaque `Color`.

This method will call the `isOpaque(HVisible)` method of an associated `HLook` if one is set. If no `HLook` is associated this method returns false.

The default return value is implementation specific and depends on the background painting mode of the given `HVisible`. The consequences of an invalid overridden value are implementation specific.

**Specified By:**

`isOpaque()` in interface `TestOpacity`

**Overrides:**

`isOpaque()` in class `HComponent`

**Returns:**

true if all the pixels with the `java.awt.Component#getBounds` method are fully opaque, i.e. its associated `HLook` guarantees that all pixels are painted in an opaque `Color`.

---

### **paint(Graphics)**

```
public void paint(java.awt.Graphics g)
```

Draws the current state of the component, by calling the `showLook(Graphics, HVisible, int)` method of the associated `HLook`. If no `HLook` is associated with the component, (i.e. the `HVisible` was created with a null `HLook` or the look has been set to null using `setLook(HLook)`) then the paint method should do nothing. This mechanism may be used for components that wish to extend `HVisible`, and override the paint method, without supporting the `HLook` interface.

**Overrides:**

java.awt.Component.paint(java.awt.Graphics) in class java.awt.Component

**Parameters:**

g - the graphics context to use for painting.

---

**setAnimateContent(Image[], int)**

```
public void setAnimateContent(java.awt.Image[] imageArray, int state)
```

Sets an array of graphical content (primarily used for animation), per state. Different (single arrays of) content can be associated with the different states of a component.

Note that the content is not copied, merely its object reference.

If the `HVisible` has an associated `HLook`, then it should repaint itself.

**Parameters:**

`imageArray` - An array of images that make up the animation. If the array is null, then any currently assigned content shall be removed for the specified state.

`state` - The state of the component for which this content should be displayed. Note that content is set on the `XXX_STATE` constants defined in `HState`, and *not* on the `XXX_STATE_BIT` constants. A `java.lang.IllegalArgumentException` will be thrown if a `STATE_BIT` rather than a `STATE` is passed.

---

**setBackgroundMode(int)**

```
public void setBackgroundMode(int mode)
```

Set the background drawing mode. The value specifies how the look should draw the background (i.e. a rectangle filling the bounds of the `HVisible`).

**Parameters:**

`mode` - one of `NO_BACKGROUND_FILL` or `BACKGROUND_FILL`

---

**setContent(Object, int)**

```
public void setContent(java.lang.Object object, int state)
```

Sets a single piece of content for this component, per state. Different (single pieces of) content can be associated with the different states of a component.

Note that the content is not copied, merely its object reference.

If the `HVisible` has an associated `HLook`, then it should repaint itself.

**Parameters:**

`object` - The content. If the content is null, then any currently assigned content shall be removed for the specified state.

`state` - The state of the component for which this content should be displayed. Note that content is set on the `XXX_STATE` constants defined in `HState`, and *not* on the `XXX_STATE_BIT` constants. A `java.lang.IllegalArgumentException` will be thrown if a `STATE_BIT` rather than a `STATE` is passed.

---

**setDefaultSize(Dimension)**

```
public void setDefaultSize(java.awt.Dimension defaultSize)
```

Set the preferred default size for this component when a layout manager is in use. If the parent Container into which the `HVisible` is placed has no layout manager this method has no effect.

Note that the size set with this method is not a *guaranteed* size; if set it will be passed to the layout manager through the `getPreferredSize(HVisible)` method. The default size of a component is to be interpreted as the area in which the component can be rendered, *excluding* look-specific borders.

Valid arguments include `NO_DEFAULT_SIZE`, and Dimensions containing `NO_DEFAULT_WIDTH` or `NO_DEFAULT_HEIGHT`

**Parameters:**

`defaultSize` - specifies the default preferred size. If this parameter is null a `java.lang.NullPointerException` will be thrown.

If this parameter or specifies a size smaller than an implementation-defined minimum size a `java.lang.IllegalArgumentException` will be thrown.

---

### **setGraphicContent(Image, int)**

```
public void setGraphicContent(java.awt.Image image, int state)
```

Sets a single piece of graphical content for this component, per state. Different (single pieces of) content can be associated with the different states of a component.

Note that the content is not copied, merely its object reference.

If the `HVisible` has an associated `HLook`, then it should repaint itself.

**Parameters:**

`image` - The content. If the content is null, then any currently assigned content shall be removed for the specified state.

`state` - The state of the component for which this content should be displayed. Note that content is set on the `XXX_STATE` constants defined in `HState`, and *not* on the `XXX_STATE_BIT` constants. A `java.lang.IllegalArgumentException` will be thrown if a `STATE_BIT` rather than a `STATE` is passed.

---

### **setHorizontalAlignment(int)**

```
public void setHorizontalAlignment(int halign)
```

Set the horizontal alignment of any state-based content rendered by an associated `HLook`. If content is not used in the rendering of this `HVisible` calls to this method shall change the current alignment mode, but this will not affect the rendered representation.

**Parameters:**

`halign` - the new horizontal alignment mode, one of `HALIGN_LEFT`, `HALIGN_CENTER`, `HALIGN_RIGHT` or `HALIGN_JUSTIFY`.

---

### **setInteractionState(int)**

```
protected void setInteractionState(int state)
```

Set the interaction state for this component. This method is provided for the use by subclasses of `HVisible` to change the interaction state of the `HVisible`. Subclasses *MUST NOT* manipulate the state by any other mechanism.

Attempts to set states which are not valid for the subclass will cause an `java.lang.IllegalArgumentException` to be thrown. See the class descriptions of each component for the definitions of which states are valid.

**Parameters:**

`state` - the interaction state for this component. A `java.lang.IllegalArgumentException` will be thrown if a `STATE_BIT` rather than a `STATE` is passed.

**See Also:**

[HState](#)

**setLook(HLook)**

```
public void setLook(HLook hlook)
```

Sets the `HLook` for this component.

**Parameters:**

`hlook` - The `HLook` that is to be used for this component. Note that this parameter may be null, in which case the component will not draw itself until a look is set.

**Throws:**

`HInvalidLookException` - If the Look is not compatible with this type of component, for example a graphic look being set on a text component, an `HInvalidLookException` is thrown. Note that `HVisible` itself will never throw this exception, but it may be thrown by a subclass which has overridden this method.

**setLookData(Object, Object)**

```
public void setLookData(java.lang.Object key, java.lang.Object data)
```

Set a look-specific private data object. Instances of `HLook` may use this method to set private data (e.g. layout hints, cached images etc.) on the `HVisible`. Use of this mechanism is an implementation option. If this mechanism is not used by an implementation, this method will have no effect and calls to `getLookData(Object)` shall return `null`.

**Parameters:**

`key` - an object which uniquely identifies the type of look for which the private data is to be retrieved. Keys need not be unique across different instances of the same look class.

`data` - a private data object, or null to remove any current object set on this `HVisible`.

**See Also:**

[HLook](#), [getLookData\(Object\)](#)

**setResizeMode(int)**

```
public void setResizeMode(int resize)
```

Set the scaling mode for scaling any state-based content rendered by an associated `HLook`. If content is not used in the rendering of this `HVisible` calls to this method shall change the current alignment mode, but this will not affect the rendered representation.

**Parameters:**

`resize` - the new scaling mode, one of `RESIZE_NONE`, `RESIZE_PRESERVE_ASPECT` or `RESIZE_ARBITRARY`

**setTextContent(String, int)**

```
public void setTextContent(java.lang.String string, int state)
```

Sets a single piece of text content for this component, per state. Different (single pieces of) content can be associated with the different states of a component. Note that unlike `setGraphicContent(Image, int)`, `setAnimateContent(Image[], int)` and

`setContent(Object, int)` , the content is copied as it is not possible to store a reference to a `java.lang.String`.

If the `HVisible` has an associated `HLook` , then it should repaint itself.

**Parameters:**

`string` - The content. If the content is null, then any currently assigned content shall be removed for the specified state.

`state` - The state of the component for which this content should be displayed. Note that content is set on the `XXX_STATE` constants defined in `HState` , and *not* on the `XXX_STATE_BIT` constants. A `java.lang.IllegalArgumentException` will be thrown if a `STATE_BIT` rather than a `STATE` is passed.

---

### **setTextLayoutManager(HTextLayoutManager)**

```
public void setTextLayoutManager(HTextLayoutManager manager)
```

Sets the text layout manager that should be used to layout the text for this component.

**Parameters:**

`manager` - the `HTextLayoutManager` to be used by this component.

---

### **setVerticalAlignment(int)**

```
public void setVerticalAlignment(int valign)
```

Set the vertical alignment of any state-based content rendered by an associated `HLook` . If content is not used in the rendering of this `HVisible` calls to this method shall change the current alignment mode, but this will not affect the rendered representation.

**Parameters:**

`valign` - the new vertical alignment mode, one of `VALIGN_TOP` , `VALIGN_CENTER` , `VALIGN_BOTTOM` or `VALIGN_JUSTIFY` .

---

### **update(Graphics)**

```
public void update(java.awt.Graphics g)
```

The `update()` method in `HVisible` overrides that in `Component` and does not clear the background of the component, it simply modifies the current `Color` of the `Graphics` object to match that of the components background `Color`, and calls the `paint()` method.

**Overrides:**

`java.awt.Component.update(java.awt.Graphics)` in class `java.awt.Component`

**Parameters:**

`g` - the graphics context to use for updating.

# Package org.havi.ui.event

Class Summary	
<b>Interfaces</b>	
<code>HActionListener</code>	The <code>HActionListener</code> interface enables the reception of <code>HActionEvent</code> events, as generated by objects implementing <code>HActionable</code> .
<code>HAdjustmentListener</code>	The <code>HAdjustmentListener</code> interface enables the reception of <code>HAdjustmentEvent</code> , as generated by objects implementing <code>HAdjustmentValue</code> .
<code>HBackgroundImageListener</code>	The listener interface for receiving events related to <code>HBackgroundImage</code> objects.
<code>HFocusListener</code>	The <code>HFocusListener</code> interface enables the reception of <code>HFocusEvent</code> events, as generated by objects implementing <code>HNavigable</code> .
<code>HItemListener</code>	The <code>HItemListener</code> interface enables the reception of <code>HItemEvent</code> , as generated by objects implementing <code>HItemValue</code> .
<code>HKeyListener</code>	The <code>HKeyListener</code> interface enables the reception of <code>HKeyEvent</code> events.
<code>HScreenConfigurationListener</code>	This listener is used to monitor when the configuration of an <code>HScreenDevice</code> is modified.
<code>HScreenLocationModifiedListener</code>	This listener is used to monitor when a component, such as an <code>HVideoComponent</code> on-screen location is modified.
<code>HTextListener</code>	The <code>HTextListener</code> interface enables the reception of <code>HTextEvent</code> , as generated by objects implementing <code>HTextValue</code> .
<b>Classes</b>	
<code>HActionEvent</code>	An <code>HActionEvent</code> event is used to interact with a component implementing the <code>HActionInputPreferred</code> interface as follows:
<code>HAdjustmentEvent</code>	An <code>HAdjustmentEvent</code> event is used to interact with a component implementing the <code>HAdjustmentValue</code> interface as follows:
<code>HBackgroundImageEvent</code>	This event informs an application that a loading operation for an <code>HBackgroundImage</code> has finished.
<code>HEventRepresentation</code>	This class is able to describe the representation of an event generator as a string, color or symbol (such as a triangle, '>', for 'play').
<code>HFocusEvent</code>	An <code>HFocusEvent</code> event is used to interact with a component implementing the <code>HNavigationInputPreferred</code> interface as follows:
<code>HItemEvent</code>	An <code>HItemEvent</code> event is used to interact with a component implementing the <code>HSelectionInputPreferred</code> interface as follows:
<code>HKeyCapabilities</code>	This class is used to describe the (basic) keyboard capabilities of the platform.
<code>HKeyEvent</code>	An <code>HKeyEvent</code> event is used to interact with a component implementing the <code>HKeyboardInputPreferred</code> interface as follows:
<code>HMouseCapabilities</code>	This class is used to describe the (basic) mouse capabilities of the platform.
<code>HRcCapabilities</code>	This class is used to describe the (basic) remote control capabilities of the platform.
<code>HRcEvent</code>	The remote control event class.

Class Summary	
<code>HScreenConfigurationEvent</code>	This event is sent to all registered <code>HScreenConfigurationListener</code> when an <code>HScreenDevice</code> modifies its configuration.
<code>HScreenDeviceReleasedEvent</code>	This event informs an application that a device for this <code>HScreen</code> has been released by an application or other entity in the system.
<code>HScreenDeviceReservedEvent</code>	This event informs that a device on this <code>HScreen</code> has been reserved by an application or other entity in the system.
<code>HScreenLocationModifiedEvent</code>	This event is generated by the system when a component is moved on-screen, rather than within a container.
<code>HTextEvent</code>	An <code>HTextEvent</code> event is used to interact with a component implementing the <code>HKeyboardInputPreferred</code> interface as follows:

# org.havi.ui.event HActionEvent

## Syntax

```
public class HActionEvent extends java.awt.event.ActionEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--java.awt.AWTEvent
|
+--java.awt.event.ActionEvent
|
+--org.havi.ui.event.HActionEvent
```

## All Implemented Interfaces:

```
java.io.Serializable
```

## Description

An `HActionEvent` event is used to interact with a component implementing the `HActionInputPreferred` interface as follows:

- An `HActionEvent` event may be sent from the HAVi system to the component to cause the component to become actioned.
- An `HActionEvent` event is sent from the component to all registered `HActionListener` whenever the component is actioned.

All interoperable HAVi components which expect to receive `HActionEvent` events must implement the `HActionInputPreferred` interface.

## Constructors

---

### HActionEvent(HActionable, int, String)

```
public HActionEvent(HActionable source, int id, java.lang.String command)
```

Constructs an `HActionEvent`.

#### Parameters:

`source` - The `HActionable` component which has been actioned.

`id` - The event id of the `HActionEvent` generated by the `HActionable` component. This is the value that will be returned by the event object's `getID` method.

`command` - A String which is used as the action command string for this event.

## Methods

---

### getActionCommand()

```
public java.lang.String getActionCommand()
```

Retrieve the action command string associated with this event.



**Overrides:**

java.awt.event.ActionEvent.getActionCommand() in class java.awt.event.ActionEvent

**Returns:**

the String which was used as the action command string when this event was constructed.

---

**getModifiers()**

```
public int getModifiers()
```

Returns any modifiers held down during this action event.

Modifiers are not used with the HAVi platform. Interoperable HAVi applications shall not use the return value of this method.

**Overrides:**

java.awt.event.ActionEvent.getModifiers() in class java.awt.event.ActionEvent

**Returns:**

an implementation-specific value.

# org.havi.ui.event

## HActionListener

### Syntax

```
public interface HActionListener extends java.awt.event.ActionListener
```

### All Superinterfaces:

java.awt.event.ActionListener, java.util.EventListener

### All Known Implementing Classes:

HEventMulticaster

### Description

The `HActionListener` interface enables the reception of `HActionEvent` events, as generated by objects implementing `HActionable`. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
-----------	-------------	---------------	------------	------------

None.

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

None.

# org.havi.ui.event HAdjustmentEvent

## Syntax

```
public class HAdjustmentEvent extends java.awt.AWTEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--java.awt.AWTEvent
|
+--org.havi.ui.event.HAdjustmentEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

An `HAdjustmentEvent` event is used to interact with a component implementing the `HAdjustmentValue` interface as follows:

- An `HAdjustmentEvent` event may be sent from the HAVi system to the component to cause a change to the value or editable mode of the component as a result of user interaction. For example, this event may be generated by a virtual keyboard user interface on a platform which lacks suitable adjustment keys.
- An `HAdjustmentEvent` event is sent from the component to all registered `HAdjustmentListener` when a change to the value or editable mode of the component occurs.

All interoperable HAVi components which expect to receive `HAdjustmentEvent` events should also implement the `HAdjustmentInputPreferred` interface. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Fields

### ADJUST\_END\_CHANGE

```
public static final int ADJUST_END_CHANGE
```

An adjustment event with this id indicates that the value of an `HAdjustmentValue` component has been finally set. This event is sent when the user causes the component to leave its editable mode. Note that it is a platform specific implementation option for such components to leave editable mode

automatically e.g. when they lose input focus. In such a case the order in which the `HFocusEvent` and `HAdjustmentEvent` are sent is platform specific.

**See Also:**

`getAdjustMode()`

---

## ADJUST\_FIRST

```
public static final int ADJUST_FIRST
```

The first integer id in the range of event ids supported by the `HAdjustmentEvent` class.

---

## ADJUST\_LAST

```
public static final int ADJUST_LAST
```

The last integer id in the range of event ids supported by the `HAdjustmentEvent` class.

---

## ADJUST\_LESS

```
public static final int ADJUST_LESS
```

An adjustment event with this id is sent whenever the value of an `HAdjustmentValue` component is decreased by one unit.

**See Also:**

`setUnitIncrement(int)`, `getUnitIncrement()`

---

## ADJUST\_MORE

```
public static final int ADJUST_MORE
```

An adjustment event with this id is sent whenever the value of an `HAdjustmentValue` component is increased by one unit.

**See Also:**

`setUnitIncrement(int)`, `getUnitIncrement()`

---

## ADJUST\_PAGE\_LESS

```
public static final int ADJUST_PAGE_LESS
```

An adjustment event with this id is sent whenever the value of an `HAdjustmentValue` component is decreased by one block.

**See Also:**

`setBlockIncrement(int)`, `getBlockIncrement()`

---

## ADJUST\_PAGE\_MORE

```
public static final int ADJUST_PAGE_MORE
```

An adjustment change event with this id is sent whenever the value of an `HAdjustmentValue` component is increased by one block.

**See Also:**

`setBlockIncrement(int)`, `getBlockIncrement()`

---

---

## ADJUST\_START\_CHANGE

```
public static final int ADJUST_START_CHANGE
```

An adjustment event with this id indicates that the value of an `HAdjustmentValue` component may be about to change. This event is sent when the user causes the component to enter its editable mode. Note that it is a platform specific implementation option for such components to enter editable mode automatically e.g. when they receive input focus. In such a case the order in which the `HFocusEvent` and `HAdjustmentEvent` are sent is platform specific.

**See Also:**

```
getAdjustMode()
```

## Constructors

---

### HAdjustmentEvent(HAdjustmentValue, int)

```
public HAdjustmentEvent(HAdjustmentValue source, int id)
```

Constructs an `HAdjustmentEvent`.

**Parameters:**

`source` - The `HAdjustmentValue` component whose value has been modified.

`id` - The event id of the `HAdjustmentEvent` generated by the `HAdjustmentValue` component. This is the value that will be returned by the event object's `getID` method.

# org.havi.ui.event HAdjustmentListener

## Syntax

```
public interface HAdjustmentListener extends java.util.EventListener
```

## All Superinterfaces:

java.util.EventListener

## All Known Implementing Classes:

HEventMulticaster

## Description

The `HAdjustmentListener` interface enables the reception of `HAdjustmentEvent`, as generated by objects implementing `HAdjustmentValue`. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Methods

### valueChanged(HAdjustmentEvent)

```
public void valueChanged(HAdjustmentEvent e)
```

Called when the value of the component has started to be changed, is changed, or has finished changing.

#### Parameters:

`e` - is the `HAdjustmentEvent` generated by the object implementing `HAdjustmentValue`.

# org.havi.ui.event HBackgroundImageEvent

## Syntax

```
public class HBackgroundImageEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.havi.ui.event.HBackgroundImageEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This event informs an application that a loading operation for an [HBackgroundImage](#) has finished. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Fields

---

### BACKGROUNDIMAGE\_FILE\_NOT\_FOUND

```
public static final int BACKGROUNDIMAGE_FILE_NOT_FOUND
```

The loading failed before attempting to load any data from the file. e.g. the file not existing or due to a badly formed or otherwise broken filename

---

### BACKGROUNDIMAGE\_FIRST

```
public static final int BACKGROUNDIMAGE_FIRST
```

Marks the first integer for the range of background image events

---

### BACKGROUNDIMAGE\_INVALID

```
public static final int BACKGROUNDIMAGE_INVALID
```

The loading failed because the data loaded is not valid. e.g. not a supported coding format for background images.

---

### BACKGROUNDIMAGE\_IOERROR

```
public static final int BACKGROUNDIMAGE_IOERROR
```

The loading failed due to an error while loading the data. e.g. the file is not accessible or loading of it was interrupted

---

### BACKGROUNDIMAGE\_LAST

```
public static final int BACKGROUNDIMAGE_LAST
```

Marks the last integer for the range of background image events

---

### BACKGROUNDIMAGE\_LOADED

```
public static final int BACKGROUNDIMAGE_LOADED
```

The loading succeeded

## Constructors

---

### HBackgroundImageEvent(Object, int)

```
public HBackgroundImageEvent(java.lang.Object source, int id)
```

Constructs a new `HBackgroundImageEvent`.

**Parameters:**

`source` - the `HBackgroundImage` which has been loaded.

`id` - the type of event (one of `BACKGROUNDIMAGE_LOADED`, `BACKGROUNDIMAGE_FILE_NOT_FOUND`, `BACKGROUNDIMAGE_IOERROR` or `BACKGROUNDIMAGE_INVALID`).

## Methods

---

### getID()

```
public int getID()
```

Returns the type for this event.

**Returns:**

the event type (one of `BACKGROUNDIMAGE_LOADED`, `BACKGROUNDIMAGE_FILE_NOT_FOUND`, `BACKGROUNDIMAGE_IOERROR` or `BACKGROUNDIMAGE_INVALID`).

---

### getSource()

```
public java.lang.Object getSource()
```

Returns the `HBackgroundImage` for which the data has been loaded.



**Overrides:**

java.util.EventObject.getSource() in class java.util.EventObject

**Returns:**

the object which has been loaded.

# org.havi.ui.event HBackgroundImageListener

## Syntax

```
public interface HBackgroundImageListener extends java.util.EventListener
```

## All Superinterfaces:

java.util.EventListener

## All Known Implementing Classes:

`HEventMulticaster`

## Description

The listener interface for receiving events related to `HBackgroundImage` objects.

## Methods

---

### **imageLoaded(HBackgroundImageEvent)**

```
public void imageLoaded(HBackgroundImageEvent e)
```

Invoked when the data for an `HBackgroundImage` has been loaded.

#### **Parameters:**

e - the event describing the loading

---

### **imageLoadFailed(HBackgroundImageEvent)**

```
public void imageLoadFailed(HBackgroundImageEvent e)
```

Invoked when loading of an `HBackgroundImage` fails.

#### **Parameters:**

e - the event describing the failure

# org.havi.ui.event HEventRepresentation

## Syntax

```
public class HEventRepresentation
    java.lang.Object
    |
    +--org.havi.ui.event.HEventRepresentation
```






## Description












This class is able to describe the representation of an event generator as a string, color or symbol (such as a triangle, '>', for 'play'). This allows an application to describe a button on an input device correctly for a given platform.

The particular text, color, or symbol can be determined by calling the `getString()`, `getColor()` or `getSymbol()` methods respectively. All available events should return a valid text representation from the `getString()` method.

If supported the six colored key events (`VK_COLORED_KEY_0` thru `VK_COLORED_KEY_5`) must also be represented by a color, i.e. the `getColor()` method must return a valid `java.awt.Color` object.

Key events may also be represented as a symbol - if the platform does not support a symbolic representation for a given event, then the application is responsible for rendering the symbol itself. The rendering of keys with a commonly known representation should follow the guidelines given here, as defined in the following table.

Event	Implied symbol	Sample
VK_GO_TO_START	Two equilateral triangles, pointing at a line to the left	
VK_REWIND	Two equilateral triangles, pointing to the left	
VK_STOP	A square	
VK_PAUSE	Two vertical lines, side by side	
VK_PLAY	One equilateral triangle, pointing to the right	

VK_FAST_FWD	Two equilateral triangles, pointing to the right	
VK_GO_TO_END	Two equilateral triangles, pointing to a line at the right	
VK_TRACK_PREV	One equilateral triangle, pointing to a line at the left	
VK_TRACK_NEXT	One equilateral triangle, pointing to a line at the right	
VK_RECORD	A circle, normally red	
VK_EJECT_TOGGLE	A line under a wide triangle which points up	
VK_VOLUME_UP	A ramp, increasing to the right, near a plus sign	
VK_VOLUME_DOWN	A ramp, increasing to the right, near a minus sign	
VK_UP	An arrow pointing up	
VK_DOWN	An arrow pointing down	
VK_LEFT	An arrow pointing to the left	

VK\_RIGHT                      An arrow pointing to the right



VK\_POWER                      A circle, broken at the top, with a vertical line in the break



The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
-----------	-------------	---------------	------------	------------

None.

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

None.

## Fields

---

### ER\_TYPE\_COLOR

```
public static final int ER_TYPE_COLOR
```

The event representation type for the current event is supported as a color.

**See Also:**

[ER\\_TYPE\\_NOT\\_SUPPORTED](#)

---

### ER\_TYPE\_NOT\_SUPPORTED

```
public static final int ER_TYPE_NOT_SUPPORTED
```

The event representation type for the current event is not supported.

The four ER\_TYPE integers describe if an input mechanism is not supported, or is described by a string, color or symbol.

The values of the four ER\_TYPE integers are required to be bitwise distinct, and the value of ER\_TYPE\_NOT\_SUPPORTED should be 0.

**See Also:**

[ER\\_TYPE\\_STRING](#), [ER\\_TYPE\\_COLOR](#), [ER\\_TYPE\\_SYMBOL](#)

---

### ER\_TYPE\_STRING

```
public static final int ER_TYPE_STRING
```

The event representation type for the current event is supported as a string.

**See Also:**

`ER_TYPE_NOT_SUPPORTED`

---

## ER\_TYPE\_SYMBOL

```
public static final int ER_TYPE_SYMBOL
```

The event representation type for the current event is supported as a symbol.

**See Also:**

`ER_TYPE_NOT_SUPPORTED`

---

## Constructors

---

### HEventRepresentation()

```
protected HEventRepresentation()
```

It is not intended that applications should directly construct `HEventRepresentation` objects.

Creates an `HEventRepresentation` object. See the class description for details of constructor parameters and default values.

This method is protected to allow the platform to override it in a different package scope.

---

## Methods

---

### getColor()

```
public java.awt.Color getColor()
```

This returns the color representation (generally used for colored soft keys) of the current event code.

**Returns:**

The color representation of the current event code, or `null` if not available.

---

### getString()

```
public java.lang.String getString()
```

Returns the text representation of the current event code.

**Returns:**

The text representation of the current event code, or `null` if not available.

---

### getSymbol()

```
public java.awt.Image getSymbol()
```

This returns an image-based representation (generally used for symbolic keys) of the current event code.

---

Note that it is platform specific whether this method will return a valid Image, in particular it is a valid implementation option to always return null. Note that for non-null Images, the size and other Image characteristics are dependent on particular manufacturer implementation.

**Returns:**

The symbolic representation of the current event code, or null if not available.

---

**getType()**

```
public int getType()
```

This returns the type of representation(s) available for the event code which this instance of `HEventRepresentation` represents.

If the event code can be represented in multiple ways, then the returned type will be the sum of the supported types, e.g. an event generated by a key with a particular font representation of an "A" in yellow might return `ER_TYPE_STRING + ER_TYPE_COLOR + ER_TYPE_SYMBOL`. Where the string representation is "A", the color representation is "yellow" and the symbol representation might be a likeness of the "A glyph" from a particular font.

---

**isSupported()**

```
public boolean isSupported()
```

This method returns true if the current event is supported by the platform.

---

**setColor(Color)**

```
protected void setColor(java.awt.Color aColor)
```

Sets the Color representation for this `HEventRepresentation`. Any previous value is overwritten.

This method is protected to allow the platform to override it in subclasses of `HEventRepresentation`. It is not intended for use by the application and conformant applications shall not use this method.

**Parameters:**

`aColor` - - the color to be associated with this event.

---

**setString(String)**

```
protected void setString(java.lang.String aText)
```

Sets the string representation for this `HEventRepresentation`. Any previous value is overwritten.

This method is protected to allow the platform to override it in subclasses of `HEventRepresentation`. It is not intended for use by the application and conformant applications shall not use this method.

**Parameters:**

`aText` - - the text string to be associated with this event.

---

**setSymbol(Image)**

```
protected void setSymbol(java.awt.Image aSymbol)
```

Sets the symbolic representation for this `HEventRepresentation`. Any previous value is overwritten.

This method is protected to allow the platform to override it in subclasses of `HEventRepresentation`. It is not intended for use by the application and conformant applications shall not use this method.

**Parameters:**

aSymbol - - the symbol image to be associated with this event.

---

**setType(int)**

```
protected void setType(int aType)
```

Sets the type of representation(s) available for the event code which this instance of `HEventRepresentation` represents. If no representation(s) are available then `aType` should be set to `ER_TYPE_NOT_SUPPORTED`. Otherwise the representation should be set to the sum of one or more of the following: `ER_TYPE_STRING`, `ER_TYPE_COLOR`, `ER_TYPE_SYMBOL`.

For example, if both string and color representations are available then the system should call this method with the parameter set to `ER_TYPE_STRING + ER_TYPE_COLOR`.

This method is protected to allow the platform to override it in subclasses of `HEventRepresentation`. It is not intended for use by the application and conformant applications shall not use this method.

**Parameters:**

aType - the type of representation(s) available for this event



# org.havi.ui.event

## HFocusEvent

### Syntax

```
public class HFocusEvent extends java.awt.event.FocusEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--java.awt.AWTEvent
|
+--java.awt.event.ComponentEvent
|
+--java.awt.event.FocusEvent
|
+--org.havi.ui.event.HFocusEvent
```

### All Implemented Interfaces:

java.io.Serializable

### Description

An `HFocusEvent` event is used to interact with a component implementing the `HNavigationInputPreferred` interface as follows:

- An `HFocusEvent` event may be sent from the HAVi system to the component to inform the component that it has gained or lost the input focus, or that it should transfer focus to another component.
- An `HFocusEvent` event is sent from the component to all registered `HFocusListener` listeners whenever the component focus status changes or focus is transferred to another component.

Note that because the underlying focus mechanism is based on AWT, focus transfer events do not guarantee that another component will actually get focus, or that the current component will lose focus.

All interoperable HAVi components which expect to receive `HFocusEvent` events must implement the `HNavigationInputPreferred` interface.

## Fields

---

### FOCUS\_TRANSFER

```
public static final int FOCUS_TRANSFER
```

An event id which indicates that the component should transfer focus to the component identified by the data returned from the `getTransferId()` method.

If a component matching the data cannot be found the component receiving this event should do nothing and not send an event to any registered `HFocusListener` listeners.

---

### HFOCUS\_FIRST

```
public static final int HFOCUS_FIRST
```

The first integer id in the range of event ids supported by the `HFocusEvent` class.

---

## HFOCUS\_LAST

```
public static final int HFOCUS_LAST
```

The last integer id in the range of event ids supported by the `HFocusEvent` class.

---

## NO\_TRANSFER\_ID

```
public static final int NO_TRANSFER_ID
```

A constant returned from the `getTransferId()` method if the event id is not `FOCUS_TRANSFER`.

## Constructors

---

### HFocusEvent(Component, int)

```
public HFocusEvent(java.awt.Component source, int id)
```

Constructs an `HFocusEvent`.

**Parameters:**

`source` - The `java.awt.Component` component which originated this event.

`id` - The event id of the `HFocusEvent` generated by the `HNavigable` component. This is the value that will be returned by the event object's `getID` method.

---

### HFocusEvent(Component, int, int)

```
public HFocusEvent(java.awt.Component source, int id, int transfer)
```

Constructs an `HFocusEvent`.

**Parameters:**

`source` - The `java.awt.Component` component which originated this event.

`id` - The event id of the `HFocusEvent` generated by the `HNavigable` component. This is the value that will be returned by the event object's `getID` method.

`transfer` - a key which maps to the component to transfer focus to, if the `id` parameter has the value `FOCUS_TRANSFER`. If the `id` parameter does not have this value `NO_TRANSFER_ID` is substituted for its value.

## Methods

---

### getTransferId()

```
public int getTransferId()
```

Returns a key which maps to the component to transfer focus to.

**Returns:**

a key which maps to the component to transfer focus to, or `NO_TRANSFER_ID` if the id of this event is not `FOCUS_TRANSFER`.

The return value of this function is used to pass key codes to an `HNavigable` to implement focus transfer for HAVi UI components.

---

**isTemporary()**

```
public boolean isTemporary()
```

Returns whether or not this focus change event is a temporary change.

**Overrides:**

java.awt.event.FocusEvent.isTemporary() in class java.awt.event.FocusEvent

**Returns:**

an implementation specific value. The HAVi UI does not use temporary focus events and interoperable applications shall not call this method.

# org.havi.ui.event

# HFocusListener

## Syntax

```
public interface HFocusListener extends java.awt.event.FocusListener
```

## All Superinterfaces:

java.util.EventListener, java.awt.event.FocusListener

## All Known Implementing Classes:

HEventMulticaster

## Description

The `HFocusListener` interface enables the reception of `HFocusEvent` events, as generated by objects implementing `HNavigable`. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
-----------	-------------	---------------	------------	------------

None.

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

None.

# org.havi.ui.event HItemEvent

## Syntax

```
public class HItemEvent extends java.awt.AWTEvent
```

```

java.lang.Object
|
+--java.util.EventObject
    |
    +--java.awt.AWTEvent
        |
        +--org.havi.ui.event.HItemEvent

```

## All Implemented Interfaces:

java.io.Serializable

## Description

An `HItemEvent` event is used to interact with a component implementing the `HSelectionInputPreferred` interface as follows:

- An `HItemEvent` event may be sent from the HAVi system to the component to change the state of the component, and the selection set held by the component.
- An `HItemEvent` event is sent from the component to all registered `HItemListener` when a change to the component state or selection set occurs.

The `HItemEvent` event class therefore provides support for selecting and clearing individual elements, selecting or clearing all elements and managing the currently focused element, i.e. the item whose selection state is toggled by some user interaction.

All interoperable HAVi components which expect to receive `HItemEvent` events should implement the `HSelectionInputPreferred` interface.

## See Also:

`HListGroup`

## Fields

---

### ITEM\_CLEARED

```
public static final int ITEM_CLEARED
```

An item event with this id is sent from the component whenever an item is removed from the selection set of an `HItemValue` component. Note that events of this type are silently ignored if they are sent to the component.

---

### ITEM\_END\_CHANGE

```
public static final int ITEM_END_CHANGE
```

An item event with this id indicates that the selection of an `HItemValue` component has been finally set. This event is sent to or from the component when the user causes the component to leave selection mode. Note that it is a platform specific implementation option for such components to

leave selection mode automatically e.g. when they lose input focus. In such a case the order in which the `HFocusEvent` and `HItemEvent` are sent is platform specific.

**See Also:**

`getSelectionMode()`

---

### ITEM\_FIRST

```
public static final int ITEM_FIRST
```

The first integer id in the range of event ids supported by the `HItemEvent` class.

---

### ITEM\_LAST

```
public static final int ITEM_LAST
```

The last integer id in the range of event ids supported by the `HItemEvent` class.

---

### ITEM\_SELECTED

```
public static final int ITEM_SELECTED
```

An item event with this id is sent from the component whenever an item is added to the selection set of an `HItemValue` component. Note that events of this type are silently ignored if they are sent to the component.

---

### ITEM\_SELECTION\_CLEARED

```
public static final int ITEM_SELECTION_CLEARED
```

An item event with this id is sent to or from the component whenever the entire selection set of an `HItemValue` component is removed.

---

### ITEM\_SET\_CURRENT

```
public static final int ITEM_SET_CURRENT
```

An item event with this id is sent to or from the component whenever the current item of an `HItemValue` component changes.

---

### ITEM\_SET\_NEXT

```
public static final int ITEM_SET_NEXT
```

An item event with this id is sent to or from the component whenever the current item of an `HItemValue` component changes to the next item in the group.

---

### ITEM\_SET\_PREVIOUS

```
public static final int ITEM_SET_PREVIOUS
```

An item event with this id is sent to or from the component whenever the current item of an `HItemValue` component changes to the previous item in the group.

---

### ITEM\_START\_CHANGE

```
public static final int ITEM_START_CHANGE
```

An item event with this id indicates that the selection of an `HItemValue` component may be about to change. This event is sent to or from the component when the user causes the component to enter selection mode. Note that it is a platform specific implementation option for such components to enter selection mode automatically e.g. when they receive input focus. In such a case the order in which the `HFocusEvent` and `HItemEvent` are sent is platform specific.

**See Also:**

`getSelectionMode()`

---

## ITEM\_TOGGLE\_SELECTED

```
public static final int ITEM_TOGGLE_SELECTED
```

An item event with this id may be sent to an `HItemValue` component to toggle the selection state of the currently focused item. Note that events of this type are never sent to listeners from the component; instead an event of type `ITEM_SELECTED` or `ITEM_CLEARED` is sent in response to an event with this id, depending on the new state of the currently focused item.

## Constructors

---

### `HItemEvent(HItemValue, int, Object)`

```
public HItemEvent(HItemValue source, int id, java.lang.Object item)
```

Constructs an `HItemEvent`.

**Parameters:**

`source` - The `HItemValue` component whose value or current item has been modified.

`id` - The event id of the `HItemEvent` generated by the `HItemValue` component. This is the value that will be returned by the event object's `getID` method.

`item` - The item which caused the change, or `null` if this information is not available. This information shall be provided if the event id is one of `ITEM_SELECTED` or `ITEM_CLEARED`.

## Methods

---

### `getItem()`

```
public java.lang.Object getItem()
```

Retrieve the item which caused this `HItemEvent`. This information is not available for the `ITEM_SELECTION_CLEARED` event id.

**Returns:**

the item which was selected or cleared, or became the current item, or `null` if this information is not available for this event.

# org.havi.ui.event

# HItemListener

## Syntax

```
public interface HItemListener extends java.util.EventListener
```

## All Superinterfaces:

java.util.EventListener

## All Known Implementing Classes:

HEventMulticaster

## Description

The `HItemListener` interface enables the reception of `HItemEvent`, as generated by objects implementing `HItemValue`. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Methods

---

### currentItemChanged(HItemEvent)

```
public void currentItemChanged(HItemEvent e)
```

Called when a change to the current item of the `HItemValue` occurs.

#### Parameters:

`e` - is the `HItemEvent` generated by the object implementing `HItemValue`.

---

### selectionChanged(HItemEvent)

```
public void selectionChanged(HItemEvent e)
```

Called when a change to the set of selected items held by the `HItemValue` occurs.

#### Parameters:

`e` - is the `HItemEvent` generated by the object implementing `HItemValue`.



# org.havi.ui.event HKeyCapabilities

## Syntax

```
public class HKeyCapabilities
    java.lang.Object
    |
    +--org.havi.ui.event.HKeyCapabilities
```

## Direct Known Subclasses:

[HRcCapabilities](#)

## Description

This class is used to describe the (basic) keyboard capabilities of the platform.

This class is not intended to be constructed by applications. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
-----------	-------------	---------------	------------	------------

None.

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

None.

## Constructors

### HKeyCapabilities()

```
protected HKeyCapabilities()
```

It is not intended that applications should directly construct [HKeyCapabilities](#) objects.

Creates an [HKeyCapabilities](#) object. See the class description for details of constructor parameters and default values.

This method is protected to allow the platform to override it in a different package scope.

## Methods

### getInputDeviceSupported()

```
public static boolean getInputDeviceSupported()
```

Determine if keyboard input functionality exists in the system. Note that this functionality may be provided through a "virtual" keyboard.

**Returns:**

true if keyboard input functionality exists in the system, false otherwise.

**See Also:**

`getInputDeviceSupported()`

---

**isSupported(int)**

```
public static boolean isSupported(int keycode)
```

Queries whether the system keyboard can generate an event of the given type. Note that this method does not distinguish between key codes which can only be generated via a virtual keyboard and key codes generated as a result of "real" key presses.

**Parameters:**

`keycode` - the keycode to query e.g. `VK_SPACE`

**Returns:**

true if events with the given key code can (ever) be generated on this system, false otherwise.

**See Also:**

`isSupported(int)`

# org.havi.ui.event HKeyEvent

## Syntax

```
public class HKeyEvent extends java.awt.event.KeyEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--java.awt.AWTEvent
|
+--java.awt.event.ComponentEvent
|
+--java.awt.event.InputEvent
|
+--java.awt.event.KeyEvent
|
+--org.havi.ui.event.HKeyEvent
```

## Direct Known Subclasses:

[HRcEvent](#)

## All Implemented Interfaces:

[java.io.Serializable](#)

## Description

An [HKeyEvent](#) event is used to interact with a component implementing the [HKeyboardInputPreferred](#) interface as follows:

- An [HKeyEvent](#) event may be sent from the HAVi system to inform the component about key-input. The source of the input may be either a real or a virtual keyboard.
- An [HKeyEvent](#) event is sent from the component to all registered [HKeyListener](#) whenever the component received input.

Note that the HAVi system should only generate KEY\_PRESSED events. Neither KEY\_TYPED nor KEY\_RELEASED events should be generated. Furthermore, the system should collapse combined events. For example, a usual Java Virtual Machine generates for the letter A three events: KEY\_PRESSED for modifier key Shift, KEY\_PRESSED for letter 'A' and KEY\_TYPED for 'A'. This should be collapsed into one single KEY\_PRESSED event with the letter 'A' and the Shift modifier set. This is to simplify the key event handling of applications.

All interoperable HAVi components which expect to receive [HKeyEvent](#) events must either implement the [HKeyboardInputPreferred](#) interface or subclass components providing the `processHKeyEvent(HKeyEvent)` method. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method

None.

**See Also:**

`processHKeyEvent (HKeyEvent)`

## Constructors

---

**HKeyEvent(Component, int, long, int, int)**

```
public HKeyEvent(java.awt.Component source, int id, long when, int modifiers, int keyCode)
```

**Deprecated.**

See explanation in `java.awt.event.KeyEvent`.

Constructs an `HKeyEvent` object with the specified source component, type, modifiers and key.

**Parameters:**

`source` - the object where the event originated.

`id` - the identifier.

`when` - the time stamp for this event.

`modifiers` - indication of any modification keys that are active for this event.

`keyCode` - the code of the key associated with this event.

---

**HKeyEvent(Component, int, long, int, int, char)**

```
public HKeyEvent(java.awt.Component source, int id, long when, int modifiers, int keyCode,  
                 char keyChar)
```

Constructs an `HKeyEvent` object with the specified source component, type, modifiers and key.

**Parameters:**

`source` - the object where the event originated.

`id` - the identifier.

`when` - the time stamp for this event.

`modifiers` - indication of any modification keys that are active for this event.

`keyCode` - the code of the key associated with this event.

`keyChar` - the character representation of the key associated with this event.

# org.havi.ui.event HKeyListener

## Syntax

```
public interface HKeyListener extends java.awt.event.KeyListener
```

## All Superinterfaces:

java.util.EventListener, java.awt.event.KeyListener

## All Known Implementing Classes:

HEventMulticaster

## Description

The `HKeyListener` interface enables the reception of `HKeyEvent` events.

Note that HAVi compliant systems should never call the methods `keyReleased(KeyEvent)` and `keyTyped(KeyEvent)` of the superinterface on an `HKeyListener`. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
-----------	-------------	---------------	------------	------------

None.

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

None.

# org.havi.ui.event HMouseCapabilities

## Syntax

```
public class HMouseCapabilities
    java.lang.Object
    |
    +--org.havi.ui.event.HMouseCapabilities
```

## Description

This class is used to describe the (basic) mouse capabilities of the platform.

This class is not intended to be constructed by applications. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Constructors

### HMouseCapabilities()

```
protected HMouseCapabilities()
```

It is not intended that applications should directly construct `HMouseCapabilities` objects.

Creates an `HMouseCapabilities` object. See the class description for details of constructor parameters and default values.

This method is protected to allow the platform to override it in a different package scope.

## Methods

### getInputDeviceSupported()

```
public static boolean getInputDeviceSupported()
```

Determine if a mouse exists in the system.

#### Returns:

true if a mouse exists in the system, false otherwise.

# org.havi.ui.event HRcCapabilities

## Syntax

```
public class HRcCapabilities extends HKeyCapabilities
```

```
java.lang.Object
|
+--HKeyCapabilities
|
+--org.havi.ui.event.HRcCapabilities
```

## Description

This class is used to describe the (basic) remote control capabilities of the platform.

This class is not intended to be constructed by applications. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Constructors

### HRcCapabilities()

```
protected HRcCapabilities()
```

It is not intended that applications should directly construct `HRcCapabilities` objects.

Creates an `HRcCapabilities` object. See the class description for details of constructor parameters and default values.

This method is protected to allow the platform to override it in a different package scope.

## Methods

### getInputDeviceSupported()

```
public static boolean getInputDeviceSupported()
```

Determine if a physical remote control exists in the system.

**Returns:**

true if a physical remote control exists in the system, false otherwise.

**See Also:**

`getInputDeviceSupported()`

---

**getRepresentation(int)**

```
public static HEventRepresentation getRepresentation(int aCode)
```

Get the `HEventRepresentation` object for a specified key event id.

**Parameters:**

`aCode` - the key event id for which the `HEventRepresentation` should be returned.

**Returns:**

an `HEventRepresentation` object for the specified key event id, or null if there is no valid representation available.

---

**isSupported(int)**

```
public static boolean isSupported(int keycode)
```

Queries whether the remote control can directly generate an event of the given type. Note that this method will return false for key codes which can only be generated on this system via a virtual keyboard.

**Parameters:**

`keycode` - the keycode to query e.g. `VK_SPACE`

**Returns:**

true if events with the given key code can be directly generated on this system via a physical remote control, false otherwise.

**See Also:**

`isSupported(int)`



# org.havi.ui.event HRcEvent

## Syntax

```
public class HRcEvent extends HKeyEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--java.awt.AWTEvent
|
+--java.awt.event.ComponentEvent
|
+--java.awt.event.InputEvent
|
+--java.awt.event.KeyEvent
|
+--HKeyEvent
|
+--org.havi.ui.event.HRcEvent
```

## All Implemented Interfaces:

```
java.io.Serializable
```

## Description

The remote control event class.

The presence or absence of these keys and their desired representation is provided by the `HRcCapabilities` class.

Note that it is an implementation option if remote control key events are repeated.

Instances of `HRcEvent` are reported through the normal `java.awt` event mechanism. Note that the reception of these events by a `java.awt.Component` is dependent on it having `java.awt.event.KeyEvent` events enabled.

Note that it is an implementation constraint that the `HRcEvent` event range should not intersect with the Java AWT key event range. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Fields

---

**RC\_FIRST**

```
public static final int RC_FIRST
```

Marks the first integer id for the range of remote control event ids.

---

**RC\_LAST**

```
public static final int RC_LAST
```

Marks the last integer id for the range of remote control event ids.

---

**VK\_BALANCE\_LEFT**

```
public static final int VK_BALANCE_LEFT
```

The 'balance left' action id - moves the audio balance to the left.

---

**VK\_BALANCE\_RIGHT**

```
public static final int VK_BALANCE_RIGHT
```

The 'balance right' action id - moves the audio balance to the right.

---

**VK\_BASS\_BOOST\_DOWN**

```
public static final int VK_BASS_BOOST_DOWN
```

The 'bass boost down' action id - decreases the audio amplifier bass boost.

---

**VK\_BASS\_BOOST\_UP**

```
public static final int VK_BASS_BOOST_UP
```

The 'bass boost up' action id - increases the audio amplifier bass boost.

---

**VK\_CHANNEL\_DOWN**

```
public static final int VK_CHANNEL_DOWN
```

The 'channel down' action id.

---

**VK\_CHANNEL\_UP**

```
public static final int VK_CHANNEL_UP
```

The 'channel up' action id.

---

**VK\_CLEAR\_FAVORITE\_0**

```
public static final int VK_CLEAR_FAVORITE_0
```

The 'clear favorite 0' action id.

---

**VK\_CLEAR\_FAVORITE\_1**

```
public static final int VK_CLEAR_FAVORITE_1
```

The 'clear favorite 1' action id.

---

**VK\_CLEAR\_FAVORITE\_2**

```
public static final int VK_CLEAR_FAVORITE_2
```

The 'clear favorite 2' action id.

---

**VK\_CLEAR\_FAVORITE\_3**

```
public static final int VK_CLEAR_FAVORITE_3
```

The 'clear favorite 3' action id.

---

**VK\_COLORED\_KEY\_0**

```
public static final int VK_COLORED_KEY_0
```

Colored key 0 action id.

Up to six colored soft keys can be included on a remote control. These are optional, and must be identified with a color. If implemented, these keys are to be oriented from left to right, or from top to bottom in ascending order.

The application can determine how many colored keys are implemented, and what colors are to be used, so that the application can match the controls, by using the `getRepresentation` method in the `HRcCapabilities` class.

**See Also:**

[VK\\_COLORED\\_KEY\\_1](#), [VK\\_COLORED\\_KEY\\_2](#), [VK\\_COLORED\\_KEY\\_3](#), [VK\\_COLORED\\_KEY\\_4](#),  
[VK\\_COLORED\\_KEY\\_5](#)

---

**VK\_COLORED\_KEY\_1**

```
public static final int VK_COLORED_KEY_1
```

Colored key 1 action id.

**See Also:**

[VK\\_COLORED\\_KEY\\_0](#)

---

**VK\_COLORED\_KEY\_2**

```
public static final int VK_COLORED_KEY_2
```

Colored key 2 action id.

**See Also:**

[VK\\_COLORED\\_KEY\\_0](#)

---

**VK\_COLORED\_KEY\_3**

```
public static final int VK_COLORED_KEY_3
```

Colored key 3 action id.

**See Also:**

[VK\\_COLORED\\_KEY\\_0](#)

---

**VK\_COLORED\_KEY\_4**

```
public static final int VK_COLORED_KEY_4
```

Colored key 4 action id.

**See Also:**

[VK\\_COLORED\\_KEY\\_0](#)

---

**VK\_COLORED\_KEY\_5**

```
public static final int VK_COLORED_KEY_5
```

Colored key 5 action id.

**See Also:**

[VK\\_COLORED\\_KEY\\_0](#)

---

**VK\_DIMMER**

```
public static final int VK_DIMMER
```

The 'device dimmer' action id adjusts illumination of the device.

This may be a toggle between two states, or a sequence through multiple states.

---

**VK\_DISPLAY\_SWAP**

```
public static final int VK_DISPLAY_SWAP
```

The 'display swap' action id - swaps displayed video sources.

---

**VK\_EJECT\_TOGGLE**

```
public static final int VK_EJECT_TOGGLE
```

The 'eject / insert media' action id.

---

**VK\_FADER\_FRONT**

```
public static final int VK_FADER_FRONT
```

The 'fader front' action id - moves the audio fader to the front.

---

**VK\_FADER\_REAR**

```
public static final int VK_FADER_REAR
```

The 'fader rear' action id - moves the audio fader to the rear.

---

**VK\_FAST\_FWD**

```
public static final int VK_FAST_FWD
```

The 'fast forward (media)' action id.

---

**VK\_GO\_TO\_END**

```
public static final int VK_GO_TO_END
```

---

The '(send media) to end position' action id.

---

### **VK\_GO\_TO\_START**

```
public static final int VK_GO_TO_START
```

The 'go (send media) to start position' action id.

---

### **VK\_GUIDE**

```
public static final int VK_GUIDE
```

The 'guide' action id - indicates a user request for a program guide (toggle).

---

### **VK\_INFO**

```
public static final int VK_INFO
```

The 'info' action id - indicates that the user has requested additional information (toggle).

---

### **VK\_MUTE**

```
public static final int VK_MUTE
```

The 'mute' action id - mute audio output

---

### **VK\_PINP\_TOGGLE**

```
public static final int VK_PINP_TOGGLE
```

The 'picture in picture toggle' action id - turns picture in picture mode on or off.

---

### **VK\_PLAY**

```
public static final int VK_PLAY
```

The 'play (media)' action id.

---

### **VK\_PLAY\_SPEED\_DOWN**

```
public static final int VK_PLAY_SPEED_DOWN
```

The 'decrease (media) play speed' action id.

---

### **VK\_PLAY\_SPEED\_RESET**

```
public static final int VK_PLAY_SPEED_RESET
```

The 'set (media) play speed to normal' action id.

---

### **VK\_PLAY\_SPEED\_UP**

```
public static final int VK_PLAY_SPEED_UP
```

The 'increase (media) play speed' action id.

---

---

**VK\_POWER**

```
public static final int VK_POWER
```

The 'device power' action id turns on or off the delegated device.

---

**VK\_RANDOM\_TOGGLE**

```
public static final int VK_RANDOM_TOGGLE
```

The 'toggle random (media) play' action id.

---

**VK\_RECALL\_FAVORITE\_0**

```
public static final int VK_RECALL_FAVORITE_0
```

The 'recall favorite 0' action id.

---

**VK\_RECALL\_FAVORITE\_1**

```
public static final int VK_RECALL_FAVORITE_1
```

The 'recall favorite 1' action id.

---

**VK\_RECALL\_FAVORITE\_2**

```
public static final int VK_RECALL_FAVORITE_2
```

The 'recall favorite 2' action id.

---

**VK\_RECALL\_FAVORITE\_3**

```
public static final int VK_RECALL_FAVORITE_3
```

The 'recall favorite 3' action id.

---

**VK\_RECORD**

```
public static final int VK_RECORD
```

The 'record (to media)' action id.

---

**VK\_RECORD\_SPEED\_NEXT**

```
public static final int VK_RECORD_SPEED_NEXT
```

The 'select next (media) record speed' action id.

---

**VK\_REWIND**

```
public static final int VK_REWIND
```

The 'rewind (media)' action id.

---

**VK\_SCAN\_CHANNELS\_TOGGLE**

```
public static final int VK_SCAN_CHANNELS_TOGGLE
```

The 'scan channels toggle' action id - turns channel scanning on or off.

---

**VK\_SCREEN\_MODE\_NEXT**

```
public static final int VK_SCREEN_MODE_NEXT
```

The 'screen mode next' action id - advances the display screen mode.

---

**VK\_SPLIT\_SCREEN\_TOGGLE**

```
public static final int VK_SPLIT_SCREEN_TOGGLE
```

The 'split screen toggle' action id - turns split screen on or off.

---

**VK\_STOP**

```
public static final int VK_STOP
```

The 'stop (media)' action id.

---

**VK\_STORE\_FAVORITE\_0**

```
public static final int VK_STORE_FAVORITE_0
```

The 'store current setting as favorite 0' action id.

---

**VK\_STORE\_FAVORITE\_1**

```
public static final int VK_STORE_FAVORITE_1
```

The 'store current setting as favorite 1' action id.

---

**VK\_STORE\_FAVORITE\_2**

```
public static final int VK_STORE_FAVORITE_2
```

The 'store current setting as favorite 2' action id.

---

**VK\_STORE\_FAVORITE\_3**

```
public static final int VK_STORE_FAVORITE_3
```

The 'store current setting as favorite 3' action id.

---

**VK\_SUBTITLE**

```
public static final int VK_SUBTITLE
```

The 'subtitle' action id - indicates a user request for subtitling (toggle).

---

**VK\_SURROUND\_MODE\_NEXT**

```
public static final int VK_SURROUND_MODE_NEXT
```

The 'surround mode next' action id - advances audio amplifier surround mode.

---

**VK\_TELETEXT**

```
public static final int VK_TELETEXT
```

The 'teletext' action id - indicates a user request for a teletext service (toggle).

---

**VK\_TRACK\_NEXT**

```
public static final int VK_TRACK_NEXT
```

The '(send media) to next track' action id.

---

**VK\_TRACK\_PREV**

```
public static final int VK_TRACK_PREV
```

The '(send media) to previous track' action id.

---

**VK\_VIDEO\_MODE\_NEXT**

```
public static final int VK_VIDEO_MODE_NEXT
```

The 'video mode next' action id - advances the display video mode.

---

**VK\_VOLUME\_DOWN**

```
public static final int VK_VOLUME_DOWN
```

The 'volume down' action id - decreases audio amplifier volume.

---

**VK\_VOLUME\_UP**

```
public static final int VK_VOLUME_UP
```

The 'volume up' action id - increases audio amplifier volume.

---

**VK\_WINK**

```
public static final int VK_WINK
```

The 'device wink' action id is used to indicate that the device should identify itself in some manner, for example, audibly or visually.

---

## Constructors

---

**HRcEvent(Component, int, long, int, int)**

```
public HRcEvent(java.awt.Component source, int id, long when, int modifiers, int keyCode)
```

**Deprecated.**

See explanation in `java.awt.event.KeyEvent`.

Constructs an `HRcEvent` object with the specified source component, type, modifiers and key.

**Parameters:**

`source` - the object where the event originated.



`id` - the identifier.

`when` - the time stamp for this event.

`modifiers` - indication of any modification keys that are active for this event.

`keyCode` - the code of the key associated with this event.

---

**HRcEvent(Component, int, long, int, int, char)**

```
public HRcEvent(java.awt.Component source, int id, long when, int modifiers, int keyCode,
                char keyChar)
```

Constructs an `HRcEvent` object with the specified source component, type, modifiers and key.

**Parameters:**

`source` - the object where the event originated.

`id` - the identifier.

`when` - the time stamp for this event.

`modifiers` - indication of any modification keys that are active for this event.

`keyCode` - the code of the key associated with this event.

`keyChar` - the character representation of the key associated with this event.

# org.havi.ui.event HScreenConfigurationEvent

## Syntax

```
public class HScreenConfigurationEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.havi.ui.event.HScreenConfigurationEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This event is sent to all registered [HScreenConfigurationListener](#) when an [HScreenDevice](#) modifies its configuration. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## See Also:

[HScreenConfigurationListener](#), [HScreenDevice](#)

## Constructors

### HScreenConfigurationEvent(Object)

```
public HScreenConfigurationEvent(java.lang.Object source)
```

Construct an [HScreenConfigurationEvent](#)

#### Parameters:

source - the [HScreenDevice](#) whose configuration changed

org.havi.ui.event

# HScreenConfigurationListener

## Syntax

```
public interface HScreenConfigurationListener extends java.util.EventListener
```

## All Superinterfaces:

java.util.EventListener

## All Known Implementing Classes:

[HEventMulticaster](#)

## Description

This listener is used to monitor when the configuration of an [HScreenDevice](#) is modified. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## See Also:

[HScreenConfigurationEvent](#), [HScreenDevice](#)

## Methods

### report(HScreenConfigurationEvent)

```
public void report(HScreenConfigurationEvent gce)
```

This method is called when the configuration of an [HScreenDevice](#) is modified.

org.havi.ui.event

# HScreenDeviceReleasedEvent

## Syntax

```
public class HScreenDeviceReleasedEvent extends org.davic.resources.ResourceStatusEvent
```

```
java.lang.Object
|
|--org.davic.resources.ResourceStatusEvent
|
|--org.havi.ui.event.HScreenDeviceReleasedEvent
```

## Description

This event informs an application that a device for this [HScreen](#) has been released by an application or other entity in the system. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Constructors

### HScreenDeviceReleasedEvent(Object)

```
public HScreenDeviceReleasedEvent(java.lang.Object source)
```

Creates an [HScreenDeviceReleasedEvent](#) object. See the class description for details of constructor parameters and default values.

**Parameters:**

`source` - the [HScreenDevice](#) which has been released

## Methods

### getSource()

```
public java.lang.Object getSource()
```

Returns the device that has been released

**Overrides:**

`org.davic.resources.ResourceStatusEvent.getSource()` in class `org.davic.resources.ResourceStatusEvent`

**Returns:**

the [HScreenDevice](#) object representing the device that has been released

# org.havi.ui.event HScreenDeviceReservedEvent

## Syntax

```
public class HScreenDeviceReservedEvent extends org.davic.resources.ResourceStatusEvent
```

```
java.lang.Object
|
+--org.davic.resources.ResourceStatusEvent
|
+--org.havi.ui.event.HScreenDeviceReservedEvent
```

## Description

This event informs that a device on this [HScreen](#) has been reserved by an application or other entity in the system. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
None.				

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
None.			

## Constructors

### HScreenDeviceReservedEvent(Object)

```
public HScreenDeviceReservedEvent(java.lang.Object source)
```

Creates an [HScreenDeviceReservedEvent](#) object. See the class description for details of constructor parameters and default values.

**Parameters:**

`source` - the [HScreenDevice](#) representing the device which has been reserved

## Methods

### getSource()

```
public java.lang.Object getSource()
```

Returns the device that has been reserved

**Overrides:**

`org.davic.resources.ResourceStatusEvent.getSource()` in class `org.davic.resources.ResourceStatusEvent`

**Returns:**

an [HScreenDevice](#) representing the device that has been reserved

# org.havi.ui.event HScreenLocationModifiedEvent

## Syntax

```
public class HScreenLocationModifiedEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.havi.ui.event.HScreenLocationModifiedEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

This event is generated by the system when a component is moved on-screen, rather than within a container. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
-----------	-------------	---------------	------------	------------

None.

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

None.

## Constructors

### HScreenLocationModifiedEvent(Object)

```
public HScreenLocationModifiedEvent(java.lang.Object source)
```

Creates an `HScreenLocationModifiedEvent` object. See the class description for details of constructor parameters and default values.

#### Parameters:

`source` - the Component whose on-screen location has been modified.

## Methods

---

**getSource()**

```
public java.lang.Object getSource()
```

Returns the Component whose on-screen location has been modified.

**Overrides:**

java.util.EventObject.getSource() in class java.util.EventObject

**Returns:**

the Component whose on-screen location has been modified.

org.havi.ui.event

# HScreenLocationModifiedListener

## Syntax

```
public interface HScreenLocationModifiedListener extends java.util.EventListener
```

## All Superinterfaces:

java.util.EventListener

## All Known Implementing Classes:

[HEventMulticaster](#)

## Description

This listener is used to monitor when a component, such as an [HVideoComponent](#) on-screen location is modified. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
-----------	-------------	---------------	------------	------------

None.

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

None.

## See Also:

[HScreenLocationModifiedEvent](#), [HVideoComponent](#)

## Methods

### report(HScreenLocationModifiedEvent)

```
public void report(HScreenLocationModifiedEvent gce)
```

This method is called when the component's on-screen location is modified.



# org.havi.ui.event HTextEvent

## Syntax

```
public class HTextEvent extends java.awt.AWTEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--java.awt.AWTEvent
|
+--org.havi.ui.event.HTextEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

An `HTextEvent` event is used to interact with a component implementing the `HKeyboardInputPreferred` interface as follows:

- An `HTextEvent` event may be sent from the HAVi system to the component to cause a change to the caret position or editable mode of the component as a result of user interaction. For example, a platform which lacks suitable caret positioning or mode switching keys may choose to generate this using a virtual keyboard user interface.
- An `HTextEvent` event is sent from the component to all registered `HTextListener` when a change to the text content, caret position or editable mode of the component occurs.

All interoperable HAVi components which expect to receive `HTextEvent` events should implement the `HKeyboardInputPreferred` interface. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
-----------	-------------	---------------	------------	------------

None.

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

None.

## Fields

### CARET\_NEXT\_CHAR

```
public static final int CARET_NEXT_CHAR
```

When a text event with this id is sent to a `HTextValue` component, then its caret position should move one character forward. If such an event is sent from a component to `HTextListener`, then it was moved.

---

### CARET\_NEXT\_LINE

```
public static final int CARET_NEXT_LINE
```

When a text event with this id is sent to a `HTextValue` component, then its caret position should move down one line. If such an event is sent from a component to `HTextListener`, then it was moved. It is widget specific, if the caret remains at the same column or at an approximate horizontal pixel position for non-fixed-width fonts.

---

### CARET\_NEXT\_PAGE

```
public static final int CARET_NEXT_PAGE
```

When a text event with this id is sent to a `HTextValue` component, then its caret position should move down to the last possible line in the visible window. If the caret position is already on the last visible line then the caret should move down so that the last visible line scrolls up to the top of the visible window. If such an event is sent from a component to `HTextListener`, then it was moved. It is widget specific, if the caret remains at the same column or at an approximate horizontal pixel position for non-fixed-width fonts.

---

### CARET\_PREV\_CHAR

```
public static final int CARET_PREV_CHAR
```

When a text event with this id is sent to a `HTextValue` component, then its caret position should move one character backward. If such an event is sent from a component to `HTextListener`, then it was moved.

---

### CARET\_PREV\_LINE

```
public static final int CARET_PREV_LINE
```

When a text event with this id is sent to a `HTextValue` component, then its caret position should move up one line. If such an event is sent from a component to `HTextListener`, then it was moved. It is widget specific, if the caret remains at the same column or at an approximate horizontal pixel position for non-fixed-width fonts.

---

### CARET\_PREV\_PAGE

```
public static final int CARET_PREV_PAGE
```

When a text event with this id is sent to a `HTextValue` component, then its caret position should move up to the first possible line in the visible window. If the caret position is already on the first visible line then the caret should move down so that the first visible line scrolls down to the bottom of the visible window. If such an event is sent from a component to `HTextListener`, then it was moved. It is widget specific, if the caret remains at the same column or at an approximate horizontal pixel position for non-fixed-width fonts.

---

### TEXT\_CARET\_CHANGE

```
public static final int TEXT_CARET_CHANGE
```

A text event with this id is sent from the component whenever the caret position of an `HTextValue` component is changed. This event will be sent only if the caret position changed in a manner not notified by the `CARET_NEXT_CHAR`, `CARET_NEXT_LINE`, `CARET_PREV_CHAR`, `CARET_PREV_LINE`, `CARET_NEXT_PAGE`, or `CARET_PREV_PAGE` events.

---

## TEXT\_CHANGE

```
public static final int TEXT_CHANGE
```

A text event with this id is sent from the component whenever the textual content of an `HTextValue` component is changed.

---

## TEXT\_END\_CHANGE

```
public static final int TEXT_END_CHANGE
```

A text event with this id indicates that the textual content of an `HTextValue` component has been finally set. This event is sent to or from the component when the user causes the component to leave its editable mode. Note that it is a platform specific implementation option for such components to leave editable mode automatically e.g. when they lose input focus. In such a case the order in which the `HFocusEvent` and `HTextEvent` are sent is platform specific.

**See Also:**

```
getEditMode()
```

---

## TEXT\_FIRST

```
public static final int TEXT_FIRST
```

The first integer id in the range of event ids supported by the `HTextEvent` class.

---

## TEXT\_LAST

```
public static final int TEXT_LAST
```

The last integer id in the range of event ids supported by the `HTextEvent` class.

---

## TEXT\_START\_CHANGE

```
public static final int TEXT_START_CHANGE
```

A text event with this id indicates that the textual content of an `HTextValue` component may be about to change. This event is sent to or from the component when the user causes the component to enter its editable mode. Note that it is a platform specific implementation option for such components to enter editable mode automatically e.g. when they receive input focus. In such a case the order in which the `HFocusEvent` and `HTextEvent` are sent is platform specific.

**See Also:**

```
getEditMode()
```

## Constructors

---

### **HTextEvent(HTextValue, int)**

```
public HTextEvent(HTextValue source, int id)
```

Constructs an `HTextEvent` .

**Parameters:**

`source` - The `HTextValue` component whose value has been modified.

`id` - The event id of the `HTextEvent` generated by the `HTextValue` component. This is the value that will be returned by the event object's `getID` method.

# org.havi.ui.event HTextListener

## Syntax

```
public interface HTextListener extends java.util.EventListener
```

## All Superinterfaces:

java.util.EventListener

## All Known Implementing Classes:

HEventMulticaster

## Description

The `HTextListener` interface enables the reception of `HTextEvent`, as generated by objects implementing `HTextValue`. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
-----------	-------------	---------------	------------	------------

None.

Default parameter values not exposed in the constructors

Description	Default value	Set method	Get method
-------------	---------------	------------	------------

None.

## Methods

### caretMoved(HTextEvent)

```
public void caretMoved(HTextEvent e)
```

Called when the caret position of an `HTextValue` component has moved.

#### Parameters:

`e` - is the `HTextEvent` generated by the object implementing `HTextValue`.

### textChanged(HTextEvent)

```
public void textChanged(HTextEvent e)
```

Called when the textual content of an `HTextValue` component has changed.

#### Parameters:

`e` - is the `HTextEvent` generated by the object implementing `HTextValue`.

## Annex W (normative): DVB-J examples

### W.1 DVB-J Application lifecycle implementation example

```

package com.random.myxlet;

/**
 * This pseudocode illustrates the kind of actions that an Xlet should
 * take in response to Xlet state change callbacks. It is primarily meant
 * to illustrate what is to be done in the paused state. An example of
 * the kind of application that might use this kind of resource-management
 * policy is a slide-show application that is used to view images. In this
 * example, the application caches only the current image when in the
 * paused state, but caches more images when in the active state.
 */

import javax.tv.Xlet;
import javax.tv.XletContext;

public class Sample implements Xlet {

    private boolean started = false;        // Latch, set true when first started
    private boolean active = false;        // Condition variable synchronized on this
    private boolean destroyed = false;     // Latch, set true when destroyed
    private XletContext context;

    public void initXlet(XletContext ctx) {
        context = ctx;
    }

    public synchronized void destroyXlet(boolean unconditional) {
        active = false;
    }

    private void manageCache() {
        for (;;) {
            synchronized(this) {
                if (destroyed) {
                    ... free any resources we hold ...;
                    cancel any image loading currently in progress;
                    return;                // Terminates thread
                }
                if (!active) {
                    free all cached images, except the one currently
                        being viewed;
                } else {                    // active
                    if (cached images remain to be loaded && an image load is not in process)
                    {
                        Start an image load. Provide the image loader with a listener that will,
                        when the image loading is complete, notify us and do this.notifyAll().
                    }
                }
            }
            wait();
        }
    }
}

```

A simple example of Xlet lifecycle is a stock ticker application that uses a back channel to retrieve stock quotes, which it displays on the viewer's television.

- a) The application manager retrieves the Xlet's code.
- b) The application manager creates an instance of the XletContext Object and initializes it for the new Xlet.
- c) The application manager initializes the Xlet by calling its `initXlet()` method and passing it the context object.

- d) The Xlet uses the context object to initialize itself and enters the `Paused` state.
- e) The application manager calls the Xlet's `startXlet()` method. The application manager assumes that the Xlet is performing its service.
- f) Upon receiving this signal, the Xlet creates a new thread that opens the back channel to retrieve the stock quotes. The Xlet is now in the `Active` state.
- g) The Xlet begins to show the stock quotes.
- h) Due to circumstances beyond the control of the Xlet, it is no longer able to retrieve updated stock quotes.
- i) The Xlet decides to continue displaying the most recent quotes it has. Note that the Xlet is still in the `Active` state.
- j) After a time, the Xlet is still unable to open the back channel. It decides that the quotes it is displaying are too old to present and that it can no longer perform its service. It chooses to take itself out of the `Active` state. It calls the `paused()` method on `XletContext` to signal this change to the application manager.
- k) Finally, the Xlet decides it no longer has any chance of performing its service, so it decides it should be terminated. It calls the `destroyed()` method on the `XletContext` to signal application manager that it has entered the `Destroyed` state. The Xlet does some final clean up.
- l) The application manager prepares the Xlet for garbage collection.

## W.2 Example of exporting an object for inter-application communication

```
public interface MyService extends java.rmi.Remote {
    public String getData() throws java.rmi.RemoteException;
}

public class MyServer extends MyService {

    private static javax.tv.xlet.XletContext ctx;

    public String getData() throws java.rmi.RemoteException {
        return "Hello from " + ctx.getXletProperty("dvb.app.id");
    }

    //
    // Called upon Xlet initialization
    //
    public static void export(XletContext ctx) {
        this.ctx = ctx;
        Remote server = new MyServer();
        org.dvb.io.ixc.IxcRegistry.bind("myserver", server);
    }

    //
    // Try to import the object that we previously exported.
    // Note that this would typically be done from a different
    // Xlet, but importing from yourself works, too.
    // Called when the Xlet is run.
    //
    public static void import(XletContext ctx) {
        String appId = ctx.getXletProperty("dvb.app.id");
        String orgId = ctx.getXletProperty("dvb.org.id");
        Remote obj;
        try {
            obj = org.dvb.io.ixc.IxcRegistry.lookup(
                "/" + appId + "/"
                + orgId + "/myserver");
            MyService r = (MyService) obj;
            System.out.println("Success " + r
                + ", " + r.getData());
        } catch (Exception ex) {
        }
    }
}
```

---

## **Annex X (normative): Test support**



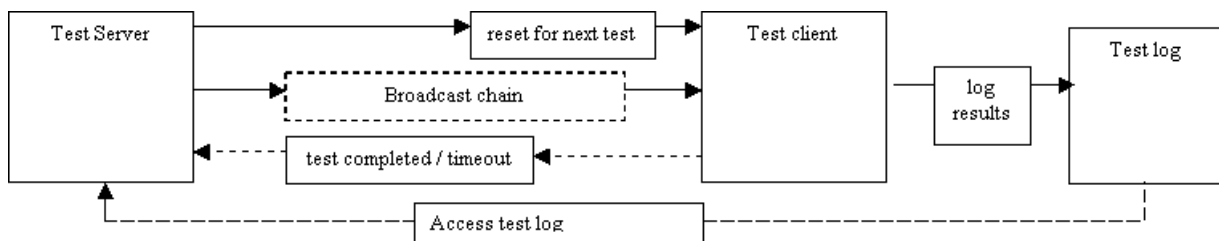
# Package org.dvb.test

## Description

### Broadcast model

In a broadcast-based conformance system, there are effectively three main entities involved in an automated test process:

1. The test server that is used to hold and initiate all of the tests.
2. The test client which runs the tests and logs the results.
3. The broadcast chain that is used to transfer applications and application data from the server to the client.



The communication order is as follows:

1. The test-server uses the "reset for next test" mechanism to set the test client into a known default state, ready to receive the test-application.
2. The test-server uses the "broadcast chain" mechanism to supply the test-application to the test client and to signal that the test-application should be executed.
3. The test-client runs the test-application.
4. The test-application either:
  - finishes within a given timelimit, the result of the test is known and shall be considered to be the value reported by the test application for the purposes of compliance.
  - Optionally, the test-client may signal to the test-server that the test-application has finished executing and that the test-client is ready to be reset in order to receive the next test-application.
  - fails to finish the test-application within a given timeout, the result of the test is unknown and shall be treated as a failure for the purposes of compliance. The test-server may treat the test-client as ready to be reset in order to receive the next application.

[Successive tests are then repeated from stage 1.]

### *"Reset for next test"*

The "reset for next test" path is used by the test server to reset the test client to receive the next test. The reset for next test API is considered to be a private implementation issue between the test-server and test-client and therefore has no public Java API implications. Note that this "communication" needs to take place prior to any application being executed. Note that the precise manner of the reset mechanism is intentionally not specified -- in the worst case, this may involve "power cycling" the test client.

***Test log***

Communication from the test client to the test log is considered as write-only access. Hence, results from successive tests cannot overwrite results from previous ones. Multiple (intermediate) results may be sent to the test log for any given test. It is recommended that all communication to the test log is synchronous.

See the DVCTest.log method for details of the proposed API and implementation issues.

***"Test completed"***

The "test completed" path is used by the test client to indicate to the test server that it has completed the previous test and is now able to accept a subsequent one. Note that this communication path is an optimisation, since direct communication from the client to the server is not actually required, e.g. the server might simply "time-out" the client, and then perform a "reset for next test" action. However, this optimisation may be important when large numbers of tests are being performed on a "capable" platform, since e.g. if a 30 second timeout is applied for 1000 test cases which typically run within say 6 seconds, then the timeout implies a typical running time of 500 minutes, i.e. ~4.5 hours --- rather than 100 minutes, i.e. ~1.5 hours.

See the DVCTest.terminate method for details of the proposed API and implementation issues.

***Access test log***

The mechanism by which the test-log is accessed is not considered in this document, this is a private mechanism, which might include reading a file from flash / RAM. Similarly, the mechanism by which results are recovered from the test log is not considered in this document, e.g. the test-log may actually reside on the test server, e.g. as in the case that results are transmitted over an IP connection.

<b>Class Summary</b>	
<b>Classes</b>	
DVCTest	The DVCTest class allows test applications to log messages during their execution and to indicate their termination condition in a platform independent manner.

# org.dvb.test

# DVBTest

## Syntax

```
public class DVBTest
```

```
java.lang.Object
```

```
|
```

```
+--org.dvb.test.DVBTest
```

## Description

The DVBTest class allows test applications to log messages during their execution and to indicate their termination condition in a platform independent manner.

A number of constants are defined in the DVBTest class and are reserved as follows

- Zero and negative values defined within the class are reserved by DVB.
- Positive return values are available for test application specific return values, which must be defined within the procedure for executing the test application as to their precise meaning as regards conformance.

## Fields

---

### FAIL

```
public static final int FAIL
```

The application executed and terminated unsuccessfully and has therefore operated in a non-conformant manner.

---

### HUMAN\_INTERVENTION

```
public static final int HUMAN_INTERVENTION
```

The application is unable to determine whether it has operated conformantly and therefore requires some human intervention to determine whether conformance has been achieved. Until the application has been checked the result of the application should be considered as non-conformant.

It is envisaged that tests returning this value may be those requiring evaluation of presented content, such as graphics, etc. Such presentation may require (subjective) human evaluation.

---

### OPTION\_UNSUPPORTED

```
public static final int OPTION_UNSUPPORTED
```

The platform does not contain the option under test and therefore the test is inapplicable, the test result should not be considered when determining the status of the platform's conformance.

---

### PASS

```
public static final int PASS
```

The application executed and terminated successfully and has therefore operated in a conformant manner.

---

## UNRESOLVED

```
public static final int UNRESOLVED
```

A setup stage necessary to execute the application failed, and hence the result of the application is unknown and therefore should be considered to have operated in a non-conformant manner.

---

## UNTESTED

```
public static final int UNTESTED
```

The application ran successfully, but the particular test was unable to execute. Hence the result of is unknown, and may require human evaluation to determine conformance.

For example, an out of disk space test may not execute within a fixed number of iterations (within a practical amount of time) for devices with large capacity storage, etc.

## Methods

---

### log(String, int)

```
public static void log(java.lang.String id, int no)
```

This method has the same behaviour, implementation options and restrictions as log(String, String) - except that it allows an integer value to be logged, rather than a String, which may prove a useful option for automating tests.

**Parameters:**

`id` - a string identifying the application (thread) that is logging the test result.

`no` - the integer value that the application wishes to be logged.

**Throws:**

`IOException` - under the same conditions as log(String, String).

---

### log(String, String)

```
public static void log(java.lang.String id, java.lang.String message)
```

This synchronous, blocking, method logs a result (intermediate result) of a test application using write-only access. The method takes both an identifier string, e.g. "Test number 1" and a message to output, e.g. "Now invoking the xletPause method...". The application is not required to open a file or network connection, per se, and the log() method is always available for writing (in principle).

The precise format of the logged message is left deliberately unspecified, implementers may choose to output compressed messages, XML documents, or other formats of their choice (obviously provided that the original information can be recovered). It is an implementation option to include additional information with each logging message, e.g. including:

- version of the specification being implemented
- compiler version and options.
- build-version
- timestamp
- date
- debug info

Messages sent using this method should "atomic", i.e. that they are not interleaved with other messages sent using the methods defined in the DVBTTest class.

## Implementation

The precise mechanism(s) by which the this method may be implemented are intentionally unspecified, implementation options might include:

- logging the message to a local file system.
- logging the message to a mounted remote file system.
- logging the message to a RAM disk, etc.
- logging the message via an RS-232 (or other serial) connection.
- logging the message to a remote host via some IP / UDP based mechanism, e.g. using a socket-based connection.

Note that the implementation of the log method may use the same or a different mechanism to that used by the terminate method.

The log method does not require any explicit initialisation on the part of the application under test. For example if messages are being stored to a file system, then the application is not required to mount / open any storage file. Similarly, if the messages are being logged via a network connection, then the application is not required to open a connection to the storage host, etc. In principle, the mechanism should always be available to accept messages.

If this method is implemented on top of some buffering mechanism, it is strongly recommended that the buffer be flushed for each occurrence of a message being logged.

## Security and implementation options

There is no Java security mechanism that is used to secure the log method.

Note that even if the log method is based on a particular implementation option, it shall be able to operate in spite of that particular implementation option itself being subject to security checks. For example, a log method implemented using the `java.net.Socket` class shall always be able to log a message from a test-application, even if the test-application is unable to directly access the `java.net.Socket` class due to security restrictions, etc.

It is an allowed implementation option to require that the test-client be put into some particular "test-mode" before any test-results are logged. This mechanism is required to reduce any inadvertent interaction due to downloaded applications accessing the test methods.

## Authoring guidelines

The log method is not intended to be accessed by downloaded applications directly, it is purely intended for the use of conformance test applications. Authors of downloaded applications should not call this method, since there may be interactions between this method and normal in-field operation of the test-client (MHP platform).

It is an allowed implementation option to require that the test-client be put into some particular "test-mode" before any test-results are logged. This mechanism is required to reduce any inadvertent interaction due to downloaded applications accessing the test methods.

It is an allowed implementation option to have a number of "test-modes" that are appropriate to different elements being conformance tested, for example, it is a valid implementation for a test-client to have a test-mode where results are stored via a serial port, and a separate test-mode where results are stored via a RAM disk. It is allowable for a conformance test to be performed with the test-client in some specific test- mode, e.g. a `java.net` test (using a serial modem) might have its test results logged to a RAM disk, to avoid interaction between test-log messages and the serial protocol.

The mechanism by which a test-client is put into a given test mode is intentionally left unspecified.

## Relationship to `java.io`

It is an implementation option to map the implementation of this method onto corresponding write method(s) of appropriate `java.io` classes. These classes may in turn be obtained, e.g. from `java.net.Socket` classes, etc.

## Parameters:

`id` - a string identifying the application (thread) that is logging the test result.

message - the message that the application wishes to be logged.

**Throws:**

`IOException` - if there is any problem in providing synchronous logging to an application. This `IOException` may be due to failure to write to a file system, inability to access a remote socket, etc. the precise causes are deliberately unspecified and are implementation dependent.

**prompt(String, int, String)**

```
public static void prompt(java.lang.String id, int controlCode, java.lang.String message)
```

This is a method is used to "approximately" synchronise a test-client and test-server, the method blocks until the test-server positively or negatively acknowledges the particular message. The intended use of this method is to remove critical timing issues from conformance tests, e.g. a conformance test to ensure that an Xlet responds to a change in broadcast signalling must first ensure that the Xlet is in a state where it is able to respond to such signalling -- since the time taken for an Xlet to achieve such a state is reliant on aspects outside of the scope of the conformance test itself (delivery bit rate, hardware and CPU capabilities of the test-client, etc.).

Messages sent using this method should "atomic", i.e. that they are not interleaved with other messages sent using the methods defined in the `DVBTest` class.

**Implementation**

The precise mechanism(s) by which the this method may be implemented are intentionally unspecified. Implementation options for sending the prompt might include:

- logging the controlCode via an RS-232 (or other serial) connection.
- logging the controlCode to a remote host via some IP / UDP based mechanism, e.g. using a socket-based connection.
- displaying the message on-screen for a (human) test operator, e.g. for systems not implementing a return channel capability.

Implementation options for receiving the acknowledgement might include:

- acknowledgement via an RS-232 (or other serial) connection.
- acknowledgement from a remote host via some IP / UDP based mechanism, e.g. using a socket-based connection.
- a (human) test operator manually acknowledging the message, e.g. for systems not implementing a return channel capability.

**Parameters:**

`id` - a string identifying the application (thread) that is sending the prompt.

`controlCode` - an integer value (unique within a given Xlet) intended for use by some automated test process (corresponding to the readable message).

`message` - a message (unique within a given Xlet) intended to be readable by a (human) test operator (corresponding to the automated controlCode).

**Throws:**

`IOException` - If there is any problem in receiving a positive acknowledgement from the test-server, then an this shall be thrown. This may be due to a negative acknowledgement from the test-server, or due to other communication based causes -- which are deliberately left unspecified.

**terminate(String, int)**

```
public static void terminate(java.lang.String id, int terminationCondition)
```

This synchronous, blocking, method logs the termination condition of a test application using write-only access. The method takes both an identifier string, e.g. "Test number 1" and a integer value to output, e.g. `org.dvb.test.DVBTest.PASS`. In addition to logging the termination condition of the test,

invoking this method also indicates that the test application has terminated its operation. Note that termination of operation does not necessarily correspond to the application being in any particular lifecycle state (as defined in the "Application Model" chapter of the MHP specification). The application is not required to open a file or network connection, per se, and the terminate() method is always available for writing (in principle).

The precise format of the termination message is left deliberately unspecified, implementers may choose to output compressed messages, XML documents, or other formats of their choice (obviously provided that the original information can be recovered). It is an implementation option to include additional information with each termination message, e.g. including:

- version of the specification being implemented
- compiler version and options.
- build-version
- timestamp
- date
- debug info

On test-clients whose implementation of the terminate() method supports external communication to its test-server, implementations of this method may optionally indicate to the test-server that the test-client can be reset by its test-server so that another test may be initiated. The precise mechanism by which this communication takes place is not specified it may be via a IP / socket, serial port, etc.

In the case of an test-client that does not support communication to its test-server, or in the case of an unsuccessful (hanging) test, or inability of this method to return (without throwing an exception) the test-server must be prepared to "time out" the application running on the test-client and then reset the test-client.

### **Implementation**

The precise mechanism(s) by which the this method may be implemented are intentionally unspecified, implementation options might include:

- storing the termination condition to a local file system.
- storing the termination condition to a mounted remote file system.
- storing the termination condition to a RAM disk, etc.
- storing the termination condition via an RS-232 (or other serial) connection.
- storing the termination condition to a remote host via some IP / UDP based mechanism, e.g. using a socket-based connection.

Messages sent using this method should "atomic", i.e. that they are not interleaved with other messages sent using the methods defined in the DVCTest class.

Note that the implementation of the terminate method may use the same or a different mechanism to that used by the log method.

The terminate method does not require any explicit initialisation on the part of the application under test. For example if termination conditions are being stored to a file system, then the application is not required to mount / open any storage file. Similarly, if the results are being logged via a network connection, then the application is not required to open a connection to the storage host, etc. In principle, the mechanism should always be available to accept termination messages.

If this method is implemented on top of some buffering mechanism, it is strongly recommended that the buffer be flushed for each occurrence of a message being logged.

### **Security and implementation options**

There is no Java security mechanism that is used to secure the terminate method.

Note that even if the terminate methods is based on a particular implementation option, it shall be able to operate in spite of that particular implementation option itself being subject to security checks. For example, a terminate method implemented using the java.net.Socket class shall always be able to log the termination condition of a test-application, even if the test-application is unable to directly access the java.net.Socket class due to security restrictions, etc.

It is an allowed implementation option to require that the test-client be put into some particular "test-mode" before any test-results are logged. This mechanism is required to reduce any inadvertent interaction due to downloaded applications accessing the test methods.

#### **Authoring guidelines**

The terminate method is not intended to be accessed by downloaded applications directly, it is purely intended for the use of conformance test applications. Authors of downloaded applications should not call this method, since there may be interactions between this method and normal in-field operation of the test-client (MHP platform).

It is an allowed implementation option to require that the test-client be put into some particular "test-mode" before any test-results are logged. This mechanism is required to reduce any inadvertent interaction due to downloaded applications accessing the test methods.

It is an allowed implementation option to have a number of "test-modes" that are appropriate to different elements being conformance tested, for example, it is a valid implementation for a test-client to have a test-mode where results are stored via a serial port, and a separate test-mode where results are stored via a RAM disk. It is allowable for a conformance test to be performed with the test-client in some specific test-mode, e.g. a java.net test (using a serial modem) might have its test results logged to a RAM disk, to avoid interaction between test-log messages and the serial protocol.

The mechanism by which a test-client is put into a given test mode is intentionally left unspecified.

#### **Relationship to java.io**

It is an implementation option to map the implementation of this method onto corresponding write method(s) of appropriate java.io classes. These classes may in turn be obtained, e.g. from java.net Socket classes, etc.

#### **Parameters:**

`id` - a string identifying the application (thread) that is terminating the test.

`terminationCondition` - the termination condition of the test application.

#### **Throws:**

`IOException` - thrown if there is any problem in terminating an application. This may be due to failure to write to a file system, inability to access a remote socket, etc. the precise causes are deliberately unspecified.



---

## **Annex Y (normative): Inter-application and Inter-Xlet communication API**

# Package org.dvb.io.ixc

## Description

Provides support for inter-application communication.

## Class Summary

### Classes

`IxcRegistry`

This is the bootstrap mechanism for obtaining references to remote objects residing in other Xlets executing on the same MHP terminal, using a URL-like syntax.

# org.dvb.io.ixc

# IxcRegistry

## Syntax

```
public class IxcRegistry

java.lang.Object
|
+--org.dvb.io.ixc.IxcRegistry
```

## Description

This is the bootstrap mechanism for obtaining references to remote objects residing in other Xlets executing on the same MHP terminal, using a URL-like syntax. The identification of a remote object is given using a syntax indicating the organisation ID and application ID:

/organisation\_id/application\_id/name

organisation\_id = the organisation ID of the Xlet, as signalled in the application\_identifier record, defined in the MHP specification.

application\_id = the application ID of the Xlet, as signalled in the application\_identifier record, defined in the MHP specification.

name = the name under which the remote object was exported.

The organisation ID and the application ID shall be encoded as a hexadecimal string, as would be accepted by `java.lang.Integer.parseInt(String s, 16)`.

When RMI is used to communicate over a network, stubs generated by a tool like `rmic` are often required. This is not necessary for inter-xlet communication initiated with `IxcRegistry`. If such stubs are present, they shall be ignored.

Similarly, network RMI objects often extend the class `server.RemoteObject`, in order to get appropriate implementations for `Object.hashCode()`, `Object.equals()`, and `Object.toString()`. Overriding `Object`'s implementation of these methods in this way is not necessary for inter-xlet communication initiated with `IxcRegistry`, although it is not harmful. Note that the class `server.RemoteObject` is not required in all MHP profiles.

## Methods

---

### **bind(XletContext, String, Remote)**

```
public static void bind(javax.tv.xlet.XletContext xc, java.lang.String name,
                       java.rmi.Remote obj)
```

Binds a remote object under a given name in the namespace of an Xlet. The name can be any valid non-null String. No hierarchical namespace exists, e.g. the names "foo" and "bar/../foo" are distinct. If the exporting xlet has been destroyed, this method may fail silently.

The object shall be made visible to other applications. A call to `bind(xc, name, obj)` is thus equivalent to a call to `bind(xc, name, obj, true)`.

#### **Parameters:**

`xc` - The context of the Xlet exporting the object.

`name` - The name identifying the object.

`obj` - The object being exported

**Throws:**

`AlreadyBoundException` - if this Xlet has previously exported an object under the given name.

`NullPointerException` - if `xc`, `name` or `obj` is null

---

**bind(XletContext, String, Remote, boolean)**

```
public static void bind(javax.tv.xlet.XletContext xc, java.lang.String name,
                       java.rmi.Remote obj, boolean external)
```

Binds a remote object under a given name in the namespace of an Xlet. The name can be any valid non-null String. No hierarchical namespace exists, e.g. the names "foo" and "bar/../foo" are distinct. If the exporting xlet has been destroyed, this method may fail silently.

**Parameters:**

`xc` - The context of the Xlet exporting the object.

`name` - The name identifying the object.

`obj` - The object being exported

`external` - Determines if this object is exported outside the scope of the current application, as defined in this specification. If set true, the object shall be visible to other applications; if false, it shall only be visible to Xlets within the same application. Note that an embedded Xlet with a different app ID than its enclosing HTML page is still considered to be the same application as that which contains the enclosing page.

**Throws:**

`AlreadyBoundException` - if this Xlet has previously exported an object under the given name.

`NullPointerException` - if `xc`, `name` or `obj` is null

**Since:**

MHP 1.1

---

**list(XletContext)**

```
public static java.lang.String[] list(javax.tv.xlet.XletContext xc)
```

Returns an array of string paths objects available in the registry. The array contains a snapshot of the names present in the registry that the current Xlet would be allowed to import using `IxcRegistry.lookup`.

**Parameters:**

`xc` - The context of the current Xlet.

**Returns:**

A non-null array of strings containing a snapshot of the path names of all objects available to the caller in this registry.

**See Also:**

[lookup\(XletContext, String\)](#)

---

**lookup(XletContext, String)**

```
public static java.rmi.Remote lookup(javax.tv.xlet.XletContext xc, java.lang.String path)
```

Returns a remote object previously exported by an Xlet that has not been destroyed. The identification of a remote object is given using a syntax indicating the organisation ID and application ID:

`/organisation_id/application_id/name`

organisation\_id = the organisation ID of the Xlet, as signalled in the application\_identifier record.

application\_id = the applicatin ID of the Xlet, as signalled in the application\_identifier record.

name = the name under which the remote object was exported.

The organisation ID and the application ID shall be encoded as a hexadecimal string, as would be accepted by `java.lang.Integer.parseInt(String s, 16)`. If the caller is not authorized to import a given object due to the security policy, then this API will behave as though the object had not been exported, that is, a `NotBoundException` shall be thrown.

**Parameters:**

`xc` - The context of the current Xlet (that is, the Xlet importing the object).

`path` - A file pathname-like string identifying the Xlet and the name of the object to be imported.

**Returns:**

A remote object

**Throws:**

`NotBoundException` - If the name is not currently bound.

`RemoteException` - If a remote stub class cannot be generated for the object being imported.

`java.lang.IllegalArgumentException` - If the name is not formatted in the syntax given above.

`NullPointerException` - if `path` is null

**rebind(XletContext, String, Remote)**

```
public static void rebind(javax.tv.xlet.XletContext xc, java.lang.String name,
    java.rmi.Remote obj)
```

Rebind the name to a new object in the context of an Xlet; replaces any existing binding. The name can be any valid non-null String. No hierarchical namespace exists, e.g. the names "foo" and "bar/../foo" are distinct. If the exporting xlet has been destroyed, this method may fail silently.

The object shall be made visible to other applications. A call to `rebind(xc, name, obj)` is thus equivalent to a call to `rebind(xc, name, obj, true)`.

**Parameters:**

`xc` - The context of the Xlet that exported the object.

`name` - The name identifying the object.

`obj` - The object being exported

**Throws:**

`NullPointerException` - if `xc`, `name` or `obj` is null

**rebind(XletContext, String, Remote, boolean)**

```
public static void rebind(javax.tv.xlet.XletContext xc, java.lang.String name,
    java.rmi.Remote obj, boolean external)
```

Rebind the name to a new object in the context of an Xlet; replaces any existing binding. The name can be any valid non-null String. No hierarchical namespace exists, e.g. the names "foo" and "bar/../foo" are distinct. If the exporting xlet has been destroyed, this method may fail silently.

**Parameters:**

`xc` - The context of the Xlet that exported the object.

`name` - The name identifying the object.

`obj` - The object being exported

`external` - Determines if this object is exported outside the scope of the current application, as defined in this specification. If set true, the object shall be visible to other applications; if false, it shall only be visible to Xlets within the same application. Note that an embedded Xlet with a different app ID than its enclosing HTML page is still considered to be the same application as that which contains the enclosing page.

**Throws:**

`NullPointerException` - if `xc`, `name` or `obj` is null

---

**unbind(XletContext, String)**

```
public static void unbind(javax.tv.xlet.XletContext xc, java.lang.String name)
```

Unbind the name.

**Parameters:**

`xc` - The context of the Xlet that exported the object to be unbound.

`name` - The name identifying the object.

**Throws:**

`NotBoundException` - if this is not currently any object exported by this Xlet under the given name.

`NullPointerException` - if `xc` or `name` is null

---

# Annex Z (informative): Services, Service Contexts and Applications in an MHP Environment

## Z.1 Introduction

This document describes the concepts that link the various parts of an MHP execution environment so that it can display a complete MHP service, including media and applications. This is really an overview to the MHP application lifecycle model, but does include some additional information.

We assume some familiarity with MHP and the JavaTV specification.

## Z.2 Basic concepts

The unit for the presentation and execution of content in the MHP specification is the service. A service in MHP represents a group of pieces of content which are intended to be presented together to the end-user. In this version of the specification, the service is the contents of a broadcast DVB service, including audio/video streams, data streams and all the service information, applications and application signalling that is being broadcast. The current service will largely be responsible for determining what media and applications are presented to the user.

Every service that gets presented by an MHP platform is presented within a service context. These form one of the foundations for the runtime environment and the execution model. A service context is an "environment" in which a service gets presented - it defines the boundaries of the service (letting the platform and applications identify which of the pieces of content that are being presented make up a given service). It also enables that service to be addressed and controlled as a single entity. A DVB-J application can call the select method on a service context (represented by `javax.tv.service.selection.ServiceContext`) and the platform will stop presenting all of the content that makes up the current service being presented by that service context and start presenting the content that makes up the new service. In this case, "content" may include one or more applications.

A service context has some major differences from a DVB-J Xlet context. It is not necessary to have one service context for every possible service that can or will get presented - one service context is needed for every service that can be presented simultaneously, but that is all. Also, a service context is not destroyed when the service within it is stopped, unlike the Xlet context for a DVB-J application. The service context exists until an application or the platform explicitly destroys it. In normal operational mode, the built-in navigator or EPG for an MHP system will create one single service context when it starts and never destroy that. MHP applications will run in that service context.

## Z.3 Presenting a service in MHP

From the MHP point of view, the content of a service can be one of two types - media or applications. Media is the simplest case and is described first.

### Z.3.1 Presenting the media components of a service

If there are several different streams of media that may get presented (such as several different video streams) then the platform uses a variety of methods to tell which streams should be used. These include user preferences and platform defaults, but will also include using service information to determine which streams get presented to the user.

For a DVB-J application, a JMF Player object is created for any media that gets presented as part of the service. These players are directly linked to the service context, and the service context can be queried to find out which JMF Players are linked with it. It is also possible for applications to create JMF Player objects directly without linkage to the applications service context.

### Z.3.2 Presenting the application components of a service

Applications are handled in a slightly different way. The lifecycle of all applications in an MHP environment is controlled by an application manager, a software entity that forms part of the MHP runtime environment. It takes its instructions on which applications to start and stop from the user, but also from information that is included in the MHP broadcast (called an "AIT", can logically be considered an extension to MPEG's PMT) and the free resources in the platform.

The information carried in the AIT (Application Information Table) not only says which applications are available, but also provides some instructions to the application manager about whether an application should be started automatically or whether an application should be killed automatically. The application manager monitors this information for changes, and creates, starts or kills applications as appropriate.

Every DVB-J application executes within an Xlet context. This is a similar concept to the applet context that a Java applet executes in, and it provides the Xlet with a link to its environment, both for accessing system properties and for telling the environment that the Xlet has changed its own state.

Since every Xlet executes as part of a DVB service, there is also a link between the Xlet and its associated service. This link is the class `javax.tv.service.selection.ServiceContextFactory`. Using methods on this class, an Xlet can lookup its service context from its xlet context. From the service context, an Xlet can discover which service it is currently running as part of. It can also register for events to be told when the service being presented in its service context changes.

The application manager must maintain a list of all the applications in a system, so that it knows which ones are currently executing. It must also know (directly or indirectly) which applications are associated with which service context, so that if the service being presented in that service context changes due to a new service being selected, it can kill the

applications that are not signalled in the new service. When a service is selected in a service context, the following steps happen in approximately this order:

- a) The platform examines the MPEG PMT for the service that has been selected (tuning first if necessary) and works out which media streams are to be presented.
- b) Any media streams currently playing are stopped and any new media streams that need to be presented are started. Any JMF players presenting the old content are stopped and destroyed, and players for the new content are created and started if necessary.
- c) The platform monitors the application information table for the new service to find out which applications should be running. Any applications that are currently running but which are not signalled in the application information table of the new service (or which are signalled as killed) are killed and the Xlet contexts of DVB-J applications associated with the old service are destroyed.
- d) For any applications which are signalled as autostart and are not currently running, the following steps are taken:
  - The platform attaches to the object carousel signalled in the application information table
  - The platform attempts to load the main application file as signalled in the application information table
  - For a DVB-J application, the platform creates an instance of the main class using the default constructor, creates an XletContext object for the application and calls the `Xlet.initXlet()` method on the newly loaded class, passing the XletContext object as a parameter. Once this call is complete, the platform calls the `Xlet.startXlet()` method.

If several applications are signalled as autostart, the platform will load and start every one in the same way. Each DVB-J application will have a different Xlet context and will execute independently, although they are all associated with the same service context.

## Z.4 Multiple service contexts in an MHP platform

The MHP specification allows the platform to have any number of service contexts, although the platform may choose to limit the number it can produce, possibly even to one. Each service context can present a different service, completely independently of the other service contexts. Operations carried out on one service context will not affect another (unless they cause tuning which prevents the contents of a service in another service context from being presented). It is even possible to display the same service in several service contexts simultaneously - this may not be very useful, but it is allowed. This may result in several instances of the same application running in at the same time in different service contexts. In practice, this is most likely to happen in future MHP terminals with multiple independent video output channels.



## Z.5 How does the platform know which services are available?

In order to select a new service, an application (or the platform) has to know three things about the service it wants to start: the original network ID, the transport stream ID and the service ID for the service in question. In the case of an application, those values may be hard-wired into the application by the developer, but this not required. The application can find out about a service in the same way that the platform can - using service information.

The platform can not know the details of every available service when it started for the first time, and so it must use another method to find out about what services it can receive. A set-top box may do the following to find this out, for instance:

- a) Scan the input (satellite, cable or some other input) to find out which transport streams are available and the physical parameters it needs to tune to them successfully.
- b) For every transport stream:
  - Tune to it
  - Use the service information API to access the service description table (SDT) and network information table (NIT) for that transport stream.
  - Use these tables to find what services are in the transport stream and the values needed to select those services
  - In the case of the platform performing an initial scan, write this information to non-volatile memory
- c) When the application has found the service it wants (or in the case of the platform performing a scan, once every transport stream has been scanned in this way), a service can be selected.

At this point, the platform has all the information it needs to start presenting services to the user. It is important to realise that this is not the only way of finding this information - other ways may be possible, depending on the MHP implementation.

# Annex AA (normative): DVB-HTML 1.0 DTD

## AA.1 DVB-HTML 1.0 DTD

### AA.1.1 DVB-HTML DTD driver

This section contains the driver for the DVB-HTML 1.0 document type implementation as an XML DTD. It relies upon XHTML module implementations defined in [Modularization of XHTML \[80\]](#) and in this annex. This driver is the DTD that shall be referenced by conformant DVB-HTML 1.0 documents.

```

<!-- ..... -->
<!--          DVB-HTML 1.0 DTD          -->
<!-- ..... -->
<!-- file: dvbhtml-1-0.dtd              -->
<!-- ..... -->

<!--          This is the DTD driver for DVB-HTML 1.0.

          The following formal public identifier shall be used to identify it:
          "-//DVB//DTD XHTML DVB-HTML 1.0//EN"

          The following URL may be used to reference this file :
          "http://www.dvb.org/mhp/dtd/dvbhtml-1-0.dtd"          -->

<!-- ..... -->
<!--          XHTML Driver Parameters          -->
<!-- ..... -->

<!ENTITY % XHTML.version "-//DVB//DTD XHTML DVB-HTML 1.0//EN" >
<!-- reserved for use with document profiles          -->
<!ENTITY % XHTML.profile "" >

<!-- ..... -->
<!--          Framework          -->
<!-- ..... -->

<!-- Tell the XHTML Framework module to use the DVB-HTML Qualified Names          -->
<!-- module as an extra qname driver          -->
<!ENTITY % xhtml-qname-extra.mod
PUBLIC "-//DVB//ENTITIES DVB-HTML Qualified Names 1.0//EN"
"dvb-qname-1.mod" >

<!-- Define the Content Model for the framework to use -->
<!ENTITY % xhtml-model.mod
PUBLIC "-//DVB//ENTITIES DVB-HTML Content Model 1.0//EN"
"dvbhtml-model-1-0.mod" >

<!-- Include bidirectional text support -->
<!ENTITY % XHTML.bidi "INCLUDE" >

<!-- Comment : Needed for tool verification purpose -->
<!ENTITY % iframe.qname "iframe" >

<!-- XHTML Inline Style Module ..... -->
<!-- needs to be included prior to the XHTML Modular Framework to be taken -->
<!-- into account          -->
<!ENTITY % xhtml-inlstyle.module "INCLUDE">
<![%xhtml-inlstyle.module;[
!ENTITY % xhtml-inlstyle.mod
PUBLIC "-//W3C//ENTITIES XHTML Inline Style 1.0//EN"
"http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-inlstyle-1.mod" >
%xhtml-inlstyle.mod;]]>

<!-- Bring in the XHTML Framework module -->
<!ENTITY % xhtml-framework.mod
PUBLIC "-//W3C//ENTITIES XHTML Modular Framework 1.0//EN"
"http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-framework-1.mod" >

```

```

%xhtml-framework.mod;

<!-- Using tables, so we need to declare this. Is also present on the Content Model -->
<!ENTITY % Table.class "| %table.qname;">

<!-- instantiate the DVB Character Entities module (for the VK_* codes). -->
<!--
NOTE 1: The content of this module is not defined in this specification
since the mappings of those DVB character entities to the character
set are implementation specific

NOTE 2: The DVB-HTML Character Entities Module is referenced from the
DVB-HTML DTDs using a local URL (see <xref AA.1.1 "DVB-HTML DTD Driver").
Hence the location from which the DTD is fetched will affect the
values given to the "VK_*" entities defined for use with the HTML
"accesskey" attribute, described in <xref to 8.5.3.1.8 "Accesskey
attribute">. Implementations may therefore choose to store these DTDs
locally and access them in an implementation-defined manner.
-->
<!ENTITY % dvb-charent.module "INCLUDE" >
<![%dvb-charent.module;[
<!ENTITY % dvb-charent.mod
SYSTEM "dvb-charent-1.mod" >
%dvb-charent.mod;]]>

<!-- ..... -->
<!-- Modules List -->
<!-- ..... -->

<!-- This DTD will accept both frameset and body elements as root element childs -->
<!ENTITY % html.content "( %head.qname; , (%frameset.qname; | %body.qname; ) )" >

<!-- XHTML Document Structure Module ..... -->
<!ENTITY % xhtml-struct.module "INCLUDE">
<![%xhtml-struct.module;[
<!ENTITY % xhtml-struct.mod
PUBLIC "-//W3C//ELEMENTS XHTML Document Structure 1.0//EN"
"http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-struct-1.mod" >
%xhtml-struct.mod;]]>

<!-- XHTML Text Module ..... -->
<!-- Fix up the blockquote content model -->
<!ENTITY % blockquote-fixed.element "INCLUDE">
<![%blockquote-fixed.element;[
<!ENTITY % blockquote.content
"( #PCDATA | %Block.mix;)*"
>
<!ENTITY % blockquote.qname "blockquote" >
<!ELEMENT %blockquote.qname; %blockquote.content; >
<!-- end of blockquote.element -->]]>
<!ENTITY % xhtml-text.module "INCLUDE">
<![%xhtml-text.module;[
<!ENTITY % xhtml-text.mod
PUBLIC "-//W3C//ELEMENTS XHTML Text 1.0//EN"
"http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-text-1.mod" >
%xhtml-text.mod;]]>

<!-- XHTML Hypertext Module ..... -->
<!ENTITY % xhtml-hypertext.module "INCLUDE">
<![%xhtml-hypertext.module;[
<!ENTITY % xhtml-hypertext.mod
PUBLIC "-//W3C//ELEMENTS XHTML Hypertext 1.0//EN"
"http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-hypertext-1.mod" >
%xhtml-hypertext.mod;]]>

<!-- XHTML Lists Module ..... -->
<!ENTITY % xhtml-list.module "INCLUDE">
<![%xhtml-list.module;[
<!ENTITY % xhtml-list.mod
PUBLIC "-//W3C//ELEMENTS XHTML Lists 1.0//EN"
"http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-list-1.mod" >

```

```

%xhtml-list.mod;]]>

<!-- XHTML Presentation Module ..... -->
<!ENTITY % xhtml-pres.module "INCLUDE">
<![%xhtml-pres.module;[
<!ENTITY % xhtml-pres.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Presentation 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-pres-1.mod" >
%xhtml-pres.mod;]]>

<!-- XHTML BDO Element Module ..... -->
<!ENTITY % xhtml-bdo.module "INCLUDE">
<![%xhtml-bdo.module;[
<!ENTITY % xhtml-bdo.mod
    PUBLIC "-//W3C//ELEMENTS XHTML BDO Element 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-bdo-1.mod" >
%xhtml-bdo.mod;]]>

<!-- XHTML Forms Module ..... -->
<!ENTITY % xhtml-form.module "INCLUDE">
<![%xhtml-form.module;[
<!ENTITY % xhtml-form.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Forms 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-form-1.mod" >
%xhtml-form.mod;]]>

<!-- XHTML Basic Tables Module ..... -->
<!ENTITY % xhtml-basic-table.module "INCLUDE">
<![%xhtml-basic-table.module;[
<!ENTITY % xhtml-basic-table.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Basic Tables 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-basic-table-1.mod">
%xhtml-basic-table.mod;]]>

<!-- XHTML Images module ..... -->
<!ENTITY % xhtml-image.module "INCLUDE">
<![%xhtml-image.module;[
<!ENTITY % xhtml-image.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Images 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-image-1.mod" >
%xhtml-image.mod;]]>

<!-- XHTML Client Side Image Map module ..... -->
<!ENTITY % xhtml-csismap.module "INCLUDE">
<![%xhtml-csismap.module;[
<!ENTITY % xhtml-csismap.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Client Side Image Maps 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-csismap-1.mod" >
%xhtml-csismap.mod;]]>

<!-- XHTML Embedded Object module ..... -->
<!ENTITY % xhtml-object.module "INCLUDE">
<![%xhtml-object.module;[
<!ENTITY % xhtml-object.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Embedded Object 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-object-1.mod" >
%xhtml-object.mod;]]>

<!-- XHTML Frames module ..... -->
<!ENTITY % xhtml-frames.module "INCLUDE">
<![%xhtml-frames.module;[
<!ENTITY % xhtml-frames.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Frames 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-frames-1.mod" >
%xhtml-frames.mod;]]>

<!-- XHTML Target module ..... -->
<!ENTITY % xhtml-target.module "INCLUDE">
<![%xhtml-target.module;[
<!ENTITY % xhtml-target.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Target 1.0//EN"

```

```

    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-target-1.mod" >
    %xhtml-target.mod;]]>

<!-- XHTML IFrame module ..... -->
<!ENTITY % xhtml-iframe.module "INCLUDE">
<![%xhtml-iframe.module;[
<!ENTITY % xhtml-iframe.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Inline Frame Element 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-iframe-1.mod" >
    %xhtml-iframe.mod;]]>

<!-- DVB-HTML Intrinsic Events Module ..... -->
<!ENTITY % dvbhtml-events.module "INCLUDE" >
<![%dvbhtml-events.module;[
<!ENTITY % dvbhtml-events.mod
    PUBLIC "-//DVB//ENTITIES DVB-HTML Intrinsic Events 1.0//EN"
    "http://www.dvb.org/mhp/dtd/dvbhtml-events-1.mod" >
    %dvbhtml-events.mod;]]>

<!-- XHTML Document Metainformation Module ..... -->
<!ENTITY % xhtml-meta.module "INCLUDE">
<![%xhtml-meta.module;[
<!ENTITY % xhtml-meta.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Metainformation 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-meta-1.mod" >
    %xhtml-meta.mod;]]>

<!-- XHTML Document Scripting Module ..... -->
<!ENTITY % xhtml-script.module "INCLUDE">
<![%xhtml-script.module;[
<!ENTITY % xhtml-script.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Scripting 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-script-1.mod" >
    %xhtml-script.mod;]]>

<!-- XHTML Document Stylesheet Module ..... -->
<!ENTITY % xhtml-style.module "INCLUDE">
<![%xhtml-style.module;[
<!ENTITY % xhtml-style.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Stylesheets 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-style-1.mod" >
    %xhtml-style.mod;]]>

<!-- XHTML Link Element Module ..... -->
<!ENTITY % xhtml-link.module "INCLUDE">
<![%xhtml-link.module;[
<!ENTITY % xhtml-link.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Link Element 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-link-1.mod" >
    %xhtml-link.mod;]]>

<!-- XHTML Base Element Module ..... -->
<!ENTITY % xhtml-base.module "INCLUDE">
<![%xhtml-base.module;[
    !ENTITY % xhtml-base.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Base Element 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-base-1.mod" >
    %xhtml-base.mod;]]>

<!-- ..... -->
<!-- XHTML Modularization modules ignored in DVB-HTML 1.0 DTD..... -->
<!-- ..... -->

<!-- XHTML Java Applet Module ..... -->
<!ENTITY % xhtml-applet.module "IGNORE">
<![%xhtml-applet.module;[
    !ENTITY % xhtml-applet.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Java Applets 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-applet-1.mod" >
    %xhtml-applet.mod;]]>

```

```

<!-- XHTML Edit Module ..... -->
<!ENTITY % xhtml-edit.module "IGNORE">
<![%xhtml-edit.module;[
    !ENTITY % xhtml-edit.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Editing Markup 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-edit-1.mod" >
    %xhtml-edit.mod;]]>

<!-- XHTML Basic Forms Module ..... -->
<!ENTITY % xhtml-basic-form.module "IGNORE">
<![%xhtml-basic-form.module;[
    !ENTITY % xhtml-basic-form.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Basic Forms 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-basic-form-1.mod" >
    %xhtml-basic-form.mod;]]>

<!-- XHTML Table Module ..... -->
<!ENTITY % xhtml-table.module "IGNORE">
<![%xhtml-table.module;[
    !ENTITY % xhtml-table.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Tables 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-table-1.mod" >
    %xhtml-table.mod;]]>

<!-- XHTML Server Side Image Map Module ..... -->
<!ENTITY % xhtml-ssismap.module "IGNORE">
<![%xhtml-ssismap.module;[
    !ENTITY % xhtml-ssismap.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Server-side Image Maps 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-ssismap-1.mod" >
    %xhtml-ssismap.mod;]]>

<!-- XHTML Intrinsic Events Module ..... -->
<!ENTITY % xhtml-events.module "IGNORE">
<![%xhtml-events.module;[
    !ENTITY % xhtml-events.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Intrinsic Events 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-events-1.mod" >
    %xhtml-events.mod;]]>

<!-- XHTML Name Identification Module ..... -->
<!ENTITY % xhtml-nameident.module "IGNORE">
<![%xhtml-nameident.module;[
    !ENTITY % xhtml-nameident.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Name Identifier 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-nameident-1.mod" >
    %xhtml-nameident.mod;]]>

<!-- XHTML Legacy Markup Module ..... -->
<!ENTITY % xhtml-legacy.module "IGNORE">
<![%xhtml-legacy.module;[
    !ENTITY % xhtml-legacy.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Legacy Markup 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-legacy-1.mod" >
    %xhtml-legacy.mod;]]>

<!-- end of DVB-HTML 1.0 DTD ..... -->
<!-- ..... -->

```

## AA.1.2 DVB-HTML DVB Intrinsic Events module

This DVB defined module adds some event attributes to the <body> and <frame> element.

```

<!-- ..... -->
<!--          DVB-HTML Intrinsic Events 1.0 Module          -->
<!-- ..... -->
<!-- file: dvbhtml-events-1.mod          -->
<!-- ..... -->

```

```

<!-- This is the DTD module for the DVB-HTML defined events.

The following formal public identifier shall be used to identify it:
    "-//DVB//ENTITIES DVB-HTML Intrinsic Events 1.0//EN"

The following URL may be used to reference this file :
    "http://www.dvb.org/mhp/dtd/dvbhtml-events-1.mod"

The namespace and the default prefix of this module are respectively
    xmlns:dvbhtml="http://www.dvb.org/mhp/"
-->

<!-- Declare PEs for the DVB Intrinsic events attributes on the body element -->
<!ENTITY % dvbhtml.body.onload.qname "%dvbhtml.pfx;onload" >
<!ENTITY % dvbhtml.body.onunload.qname "%dvbhtml.pfx;onunload" >
<!ENTITY % dvbhtml.body.ondvbdomstable.qname "%dvbhtml.pfx;ondvbdomstable" >

<!-- Declare PEs for the DVB-HTML Intrinsic events attributes on the
frameset element (only used in the DVB-HTML Frameset DTD)
-->
<!ENTITY % dvbhtml.frameset.onload.qname "%dvbhtml.pfx;onload" >
<!ENTITY % dvbhtml.frameset.onunload.qname "%dvbhtml.pfx;onunload" >
<!ENTITY % dvbhtml.frameset.ondvbdomstable.qname "%dvbhtml.pfx;ondvbdomstable" >

<!-- additional attributes on body element -->
<!ATTLIST %body.qname;
    %dvbhtml.body.onload.qname      %Script.datatype;      #IMPLIED
    %dvbhtml.body.onunload.qname    %Script.datatype;      #IMPLIED
    %dvbhtml.body.ondvbdomstable.qname %Script.datatype;    #IMPLIED
>

<!-- end of dvbhtml-events-1.mod ..... -->
<!-- ..... -->

```

### AA.1.3 DVB-HTML Qualified Names module

This DVB-defined module is declared in the DVB-HTML DTD driver and instantiated through the XHTML Modular Framework. It defines parameter entities to support namespace-qualified names. In particular, it defines namespace declarations and name prefixing rules. The actual namespace defined by this module is provided by the following URI: "http://www.dvb.org/mhp/" and the default prefix by the string "dvbhtml".

It is defined below:

```

<!-- ..... -->
<!-- DVB-HTML 1.0 QName Module -->
<!-- ..... -->
<!-- file: dvbhtml-qname-1.mod -->
<!-- ..... -->

<!--
The following formal public identifier shall be used to identify it:
    "-//DVB//ENTITIES DVB-HTML Qualified Names 1.0//EN"

The following URL may be used to reference this file :
    "http://www.dvb.org/mhp/dtd/dvbhtml-qname-1.mod"
-->

<!-- Bring in the XHTML datatypes, since the URI.datatype parameter entity -->
<!-- is used for declaring the xmlns attributes. -->
<!ENTITY % dvb-datatypes.mod
    PUBLIC "-//W3C//ENTITIES XHTML Datatypes 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-datatypes-1.mod" >
%dvb-datatypes.mod;

<!-- Declare the default value for prefixing of this module's attributes -->
<!-- Note that the NS.prefixed will get overridden by the XHTML Framework -->
<!-- or by a document instance. -->

```

```

<!ENTITY % NS.prefixed "IGNORE" >
<!ENTITY % dvbhtml.prefixed "%NS.prefixed;" >

<!-- Declare the actual namespace of this module -->
<!ENTITY % dvbhtml.xmlns "http://www.dvb.org/mhp/" >

<!-- Declare the default prefix for this module -->
<!ENTITY % dvbhtml.prefix "dvbhtml" >

<!-- Declare the prefix and any prefixed namespaces that are required by -->
<!-- this module. -->
<![%dvbhtml.prefixed;[
<!ENTITY % dvbhtml.pfx "%dvbhtml.prefix;" >
<!ENTITY % dvbhtml.xmlns.extra.attrib
    "xmlns:%dvbhtml.prefix; %URI.datatype; #FIXED '%dvbhtml.xmlns;' " >
]]>
<!ENTITY % dvbhtml.pfx "" >

<!-- Declare a Parameter Entity (PE) that defines any external namespaces -->
<!-- that are used by this module -->
<!ENTITY % dvbhtml.xmlns.extra.attrib "" >

<!-- Declare a PE that defines the xmlns attributes for use by dvbhtml. -->
<![%dvbhtml.prefixed;[
<!ENTITY % dvbhtml.xmlns.attrib
    "xmlns:%dvbhtml.prefix; %URI.datatype; #FIXED i%dvbhtml.xmlns;i
    %dvbhtml.xmlns.extra.attrib;"
>
]]>
<!ENTITY % dvbhtml.xmlns.attrib
    "xmlns %URI.datatype; #FIXED i%dvbhtml.xmlns;i
    %dvbhtml.xmlns.extra.attrib;"
>

<!-- Make sure that the dvbhtml namespace attributes are included on the -->
<!-- XHTML attribute set -->
<![%NS.prefixed;[
<!ENTITY % XHTML.xmlns.extra.attrib
    "%dvbhtml.xmlns.attrib;" >
]]>
<!ENTITY % XHTML.xmlns.extra.attrib "" >

<!-- end of dvbhtml-gname-1.mod ..... -->
<!-- ..... -->

```

## AA.1.4 DVB-HTML Content Model module

The DVB-HTML content model module works together with the DVB-HTML 1.0 DTD driver to customize the XHTML module implementations to the document type's specific requirements. This module is instantiated by the XHTML modular framework module.

It is defined below:

```

<!-- ..... -->
<!--      DVB-HTML 1.0 Document Model Module      -->
<!-- ..... -->
<!-- file: dvbhtml-model-1-0.mod      -->
<!-- ..... -->

<!-- This is the DTD module for the DVB-HTML Document Model.

    The following formal public identifier shall be used to identify it:
        PUBLIC "-//DVB//ENTITIES DVB-HTML Document Model 1.0//EN"

    The following URL may be used to reference this file :
        "http://www.dvb.org/mhp/dtd/dvbhtml-model-1-0.mod"      -->

<!-- Document Model

```



This module describes the groupings of elements that make up common content models for DVB-HTML elements.

-->

```

<!-- ..... Optional Elements in head ..... -->

<!ENTITY % HeadOpts.mix "(
    | %script.qname;
    | %style.qname;
    | %meta.qname;
    | %link.qname;
    | %object.qname;)*" >

<!-- ..... Miscellaneous Elements ..... -->

<!-- script and noscript are used to contain scripts -->

<!ENTITY % Misc.extra " | %script.qname;
    | %noscript.qname; " >

<!-- These elements are neither block nor inline, and can
      essentially be used anywhere in the document body. -->
<!ENTITY % Misc.class " %Misc.extra; " >

<!-- ..... Inline Elements ..... -->

<!ENTITY % InlStruct.class " | %br.qname;
    | %span.qname; " >

<!ENTITY % InlPhras.class " | %em.qname;
    | %strong.qname;
    | %dfn.qname;
    | %code.qname;
    | %samp.qname;
    | %kbd.qname;
    | %var.qname;
    | %cite.qname;
    | %abbr.qname;
    | %acronym.qname;
    | %q.qname; " >

<!ENTITY % InlPres.class " | %tt.qname;
    | %i.qname;
    | %b.qname;
    | %big.qname;
    | %small.qname;
    | %sub.qname;
    | %sup.qname; " >

<!ENTITY % I18n.class " | %bdo.qname; " >

<!ENTITY % Anchor.class " | %a.qname; " >

<!ENTITY % InlSpecial.class " | %img.qname;
    | %map.qname;
    | %object.qname; " >

<!ENTITY % Control.class " | %input.qname;
    | %select.qname;
    | %textarea.qname;
    | %label.qname;
    | %button.qname; " >

<!ENTITY % Inline.extra " | %iframe.qname; " >

<!-- %Inline.class; includes all inline elements, used as a component in mixes -->
<!ENTITY % Inline.class " %InlStruct.class;
    %InlPhras.class;
    %InlPres.class;
    %I18n.class;

```

```

        %Control.class;
                %Anchor.class;
                %InlSpecial.class;
                %Inline.extra;" >

<!-- %InlNoAnchor.class; includes all non-anchor inline elements,
used as a component in mixes -->

<!ENTITY % InlNoAnchor.class      " %InlStruct.class;
                %InlPhras.class;
                %InlPres.class;
                %I18n.class;
        %Control.class;
                %InlSpecial.class;
                %Inline.extra;" >

<!-- %Inline-noa.mix; includes all non-anchor inlines -->
<!ENTITY % InlNoAnchor.mix        " %InlNoAnchor.class;
                %Misc.class;">

<!-- %Inline.mix; includes all inline elements, including %Misc.class; -->
<!ENTITY % Inline.mix              " %Inline.class;
                %Misc.class;" >

<!-- ..... Block Elements ..... -->

<!ENTITY % Heading.class          " %h1.qname;
        | %h2.qname;
        | %h3.qname;
        | %h4.qname;
        | %h5.qname;
        | %h6.qname; " >

<!ENTITY % List.class             " %ul.qname;
        | %ol.qname;
        | %dl.qname; " >

<!ENTITY % BlkStruct.class        " %p.qname;
        | %div.qname; " >

<!ENTITY % BlkPhras.class         " | %pre.qname;
        | %blockquote.qname;
        | %address.qname; " >

<!ENTITY % BlkPres.class          " | %hr.qname; " >

<!ENTITY % BlkNoTable.extra       " | %form.qname;
        | %fieldset.qname; " >

<!ENTITY % Table.class           " | %table.qname; " >

<!ENTITY % Block.extra            "%Table.class;
        | %form.qname;
        | %fieldset.qname; " >

<!ENTITY % BlkNoTable.class       "%BlkStruct.class;
                %BlkPhras.class;
                %BlkPres.class;
                %BlkNoTable.extra;" >

<!-- %Block.class; includes all block elements, used as a component in mixes -->
<!ENTITY % Block.class            "%BlkStruct.class;
                %BlkPhras.class;
                %BlkPres.class;
                %Block.extra;" >

<!-- %Block.mix; includes all block elements plus %Misc.class; -->
<!ENTITY % Block.mix              "%Heading.class;

```

```
        | %List.class;  
        | %Block.class;  
        | %Misc.class;" >  
  
<!-- ..... All Content Elements ..... -->  
  
<!ENTITY % FlowNoTable.mix      "%Heading.class;  
        | %List.class;  
        | %BlkNoTable.class;  
        | %Inline.class;  
        | %Misc.class;" >  
  
<!-- %Flow.mix; includes all text content, block and inline -->  
<!ENTITY % Flow.mix            "%Heading.class;  
        | %List.class;  
        | %Block.class;  
        | %Inline.class;  
        | %Misc.class;" >  
  
<!-- ..... -->  
<!-- end of dvbhtml-model-1-0.mod ..... -->  
<!-- ..... -->
```

## Annex AB (normative): DVB HTML StyleSheet

In the absence of any other styling information for a page, the support application shall render using the following default stylesheet:

```

/*
 * Since MHP implementations may add element types, this rule prevents
 * any such elements from being displayed by default. The following
 * rule restores the default "inline" value to HTML element types which
 * should be displayed inline.
 */
*
      { display: none }
style, meta, link,
script, noscript,
br, span,
em, strong, dfn,
code, samp, kbd,
var, cite, abbr,
acronym, q,
tt, i, d, big,
small, sub, sup,
bdo, a,
img, map, object,
input, select,
textarea, label,
button      { display: inline }

address,
blockquote,
body, dd, div,
dl, dt, fieldset,
form, frame,
frameset, h1, h2,
h3, h4, h5, h6,
hr, noframes, ol,
p, pre, ul   { display: block }
object      { display: inline }
li          { display: list-item }
head        { display: none }
table       { display: table }
tr          { display: table-row }

/*
 * NOTE: Removed for DVB-HTML, as these elements and these values of display
 * property are not supported.
 *
 * thead      { display: table-header-group }
 * tbody     { display: table-row-group }
 * tfoot     { display: table-footer-group }
 * col       { display: table-column }
 * colgroup  { display: table-column-group }
 */
td, th      { display: table-cell }
caption     { display: table-caption }
th          { font-weight: bolder; text-align: center }
caption     { text-align: center }

/*
 * NOTE: The padding here and the initial block below (in the @viewport
 * rule) is defined such that the initial block matches the "Safe Action
 * Area" and the content area of the BODY matches the "Safe Title Area".
 * The Safe Action Area is the centre 90% of the screen; the Safe Title
 * Area is the centre 80%. Percentages are not used to specify the body
 * padding because both axes are interpreted with respect to the *width*
 * of the containing block. Therefore the padding below is in pixels.
 *
 * NOTE: The sample HTML 4.0 stylesheet in the CSS2 specification, derived
 * from a survey of existing User Agents, has "line-height: 1.33". However,
 * the errata (as of 19980512) reduce this to 1.12em.

```

```

*/
body      { font-family: Tiresias; font-size: 26pt;
           padding: 29px 36px; line-height: 1.12em;
           opacity: 0;
           background-color: rgb(0, 0, 0);
           color: rgb(255, 255, 255); }

/*
* NOTE: The margins here are chosen to be 26pt/18px (according to the
* information on Tiresias in TAM232r16). They are
* expressed in "em" units so overriding stylesheets can change just the
* font-size and have the margins scaled.
*/
h6, h4, h2      { text-transform: uppercase }
h6            { font-size: 24pt; margin: 1.08em 0 }
h5, h4        { font-size: 26pt; margin: 1em 0 }
h3, h2        { font-size: 31pt; margin: 0.84em 0 }
h1           { font-size: 36pt; margin: 0.72em 0 }

/*
* NOTE: The vertical margin here is left at 1.33em despite the CSS 2 errata
* mentioned above.
*/
blockquote, dl, fieldset, form, ol, p, ul { margin: 1.33em 0 }
h1, h2, h3, h4, h5, h6,
strong       { font-weight: bolder }
blockquote   { margin-left: 1.5em; margin-right: 1.5em }
address, cite,
em, i, var   { font-style: italic }
pre, code, kbd,
samp        { font-family: monospace }
pre         { white-space: pre }
big         { font-size: larger }
small, sub, sup { font-size: smaller }
sub         { vertical-align: sub }
sup         { vertical-align: super }
hr          { border: 1px solid }
ol, ul, dd  { margin-left: 1.5em }
ol          { list-style-type: decimal }
ol ul, ul ol,
ul ul, ol ol { margin-top: 0; margin-bottom: 0 }

br:before   { content: "\A" }

abbr, acronym { letter-spacing: 0.1em }

:link       { color: rgb(0, 127, 255) } /* opaque blue */
:focus      { color: rgb(0, 127, 255); } /* opaque blue */
           outline: 2px solid rgb(255, 0, 127) } /* opaque blue */
:active     { color: rgb(255, 31, 0); } /* opaque red */
           outline: 2px solid rgb(255, 31, 0) } /* opaque red */
:visited    { color: rgb(191, 0, 127) } /* opaque violet */

/* Begin bidirectionality settings (do not change) */
bdo[dir="ltr"] { direction: ltr; unicode-bidi: bidi-override }
bdo[dir="rtl"] { direction: rtl; unicode-bidi: bidi-override }

*[dir="ltr"]   { direction: ltr; unicode-bidi: embed }
*[dir="rtl"]   { direction: rtl; unicode-bidi: embed }

/* Elements that are block-level in HTML4 */
/*
* NOTE: Removed for DVB-HTML, as these elements and these values of display
* property are not supported: thead, tbody, tfoot, col, colgroup
*/
address, blockquote, body, dd, div, dl, dt, fieldset,
form, frame, frameset, h1, h2, h3, h4, h5, h6, iframe,
noscript, noframes, object, ol, p, ul,
hr, pre, li, table, tr, td, th, caption { unicode-bidi: embed }
/* End bidi settings */

```

```
@viewport {  
  /* scene: defaults to rect(0%, 0%, 100%, 100%), full screen */  
  /* horizontal-resolution: defaults to 720px; */  
  /* vertical-resolution: 576px; */  
  /* By default, stay within Safe Action Area (centre 90%) */  
  /* initial: (36px, 29px, 648px, 518px); */  
  initial: rect(10%, 10%, 90%, 90%);  
  type: none;  
  background: rgb(0, 0, 0);  
  area: screen;  
}
```

---

## Annex AC (normative): ECMAScript Binding

This appendix contains the complete [ECMAScript \[81\]](#) binding for the DVB HTML Document Object Model ECMAScript Language Binding.

### AC.1 ECMAScript language binding

Since ECMAScript is a prototype-based language, the DOM implementation using this language can add mechanisms which can't be defined using the IDL language. This section defines two sets of functionalities that an ECMAScript DOM DVB implementation shall support.

For other considerations concerning the ECMAScript language binding, please refer to the appendix D of [HTML 4 \[90\]](#).

#### AC.1.1 Shortcuts to access objects

The IDL interfaces defined in section 8.11.4.6, "[DVB-HTML element related interfaces](#)" on page 140 provide access to some objects using DVB HTML collections. For example, the third form element of a document can be accessed by using the following instruction, where `document` is the name of the object representing the DVB HTML document :

```
document.forms[2]
```

In this perspective, each object shall add a new property for each object that it encapsulates. The name of the property shall be the identifier of the corresponding encapsulated object (i.e. the value of the `id` attribute inherited by the `DVBHTMLInputElement` interface). The document will fail to parse if the `id` of an element would rename a property of the host object. Setting a property on the ECMAScript GlobalObject will override the shortcut of the same name.

This property only exists if the element has an `id` attribute. This shall be applied for the following interfaces:

- `DVBHTMLAnchorElement`,
- `DVBHTMLMapElement`,
- `DVBHTMLAreaElement`,
- `DVBHTMLButtonElement`,
- `DVBHTMLFrameElement`,
- `DVBHTMLFramesetElement`,
- `DVBHTMLImageElement`,
- `DVBHTMLInputElement`,
- `DVBHTMLOptionElement`,
- `DVBHTMLSelectElement`,
- `DVBHTMLTextAreaElement`.

For example, if a document contains a form whose identifier is `myForm`, the form can be accessed using the following equivalent instructions :

```
document.myForm  
document["myForm"]  
myForm // since document is the global object
```

Subsequently, if the form contains a button identified by the `myButton` string, the button can be accessed using :

```
document.myForm.myButton
```

## AC.1.2 Grouping the objects of a form

In DVB HTML, the following elements of a form may belong to different groups: button, input, select, textarea. The `name` attribute of those elements identifies this group. The `DVBHTMLFormElement` interface doesn't represent this relation. Instead, all the elements included in the form are accessed by the `elements` collection. To determine the elements belonging to the same group, an application can list all the elements, looking for the ones with the `name` attribute equals to a particular group name.

A DOM DVB HTML implementation shall provide an easier access by integrating a property in a form element for each group included in this form. The property is a table containing all the elements of the group. The name of the property shall be the `name` attribute of the elements which corresponds to the name of the group. For example, if the DVB HTML document contains a form identified by `myForm`, with two elements having their `name` attribute set to `myGroup`, these buttons will be accessed using the following instructions :

```
document.myForm.myGroup[0]
document.myForm.myGroup[1]
```

The rule defined in the previous section also applies. Thus, if the two elements are respectively identified by `myInput1`, and `myInput2`, they can also be accessed by the following instructions :

```
document.myForm.myInput1
document.myForm.myInput2
```

Methods that attempt to set read-only attributes shall cause the `DOMException` with the error code `NO_MODIFICATION_ALLOWED_ERR` to be raised.

## AC.2 The DVB-HTML host objects

### AC.2.1 Object `DVBHTMLCollection`

The `DVBHTMLCollection` object has the following properties:

**length:** This read-only property is of type `Number`.

The `DVBHTMLCollection` object has the following methods:

**item(index):** This method returns a `Node` object.

The **index** parameter is of type `Number`.

Note: This object can also be dereferenced using square bracket notation (e.g. `obj[1]`).

Dereferencing with an integer index is equivalent to invoking the `item` method with that index.

**namedItem(name):** This method returns a `Node` object.

The **name** parameter is of type `String`.

Note: This object can also be dereferenced using square bracket notation (e.g. `obj["foo"]`). Dereferencing using a string index is equivalent to invoking the `namedItem` method with that name.

NOTE: In DVB HTML names are not used for identification purposes, so only `Nodes` with `id` attributes will appear in a `DVBHTMLCollection` object.

See:

- 8.11.4.5.1, "DVB-HTMLCollection Interface" on page 137

### AC.2.2 Object `DVBHTMLDocument`

`DVBHTMLDocument` has the all the properties and methods of the `Document` object as well as the properties and methods defined below.



The DVBHTMLDocument object has the following properties:

**title:** This property is of type String.

**referrer:** This read-only property is of type String.

**domain:** This read-only property is of type String.

**URL:** This read-only property is of type String.

**body:** This property is a DVBHTMLCollection object.

**images:** This read-only property is a DVBHTMLCollection object.

**links:** This read-only property is a DVBHTMLCollection object.

**forms:** This read-only property is a DVBHTMLCollection object.

**anchors:** This read-only property is a DVBHTMLCollection object.

**cookie:** This property is of type String.

The DVBHTMLDocument object has the following methods:

**open():** This method has no return value. The ECMAScript context which opened the new document cannot be destroyed until it calls document.close().

**close():** This method has no return value. Deletes the ECMAScript context that opened the document.

**write(text):** This method has no return value.

The **text** parameter is of type String.

**writeln(text):** This method has no return value.

The **text** parameter is of type String.

See:

- [8.11.4.5.2, "DVBHTMLDocument Interface" on page 138](#)

### AC.2.3 Object DVBHTMLCollection

DVBHTMLCollection has all the properties and methods of the Element object as well as the properties and methods defined below.

The DVBHTMLCollection object has the following properties:

**id:** This property is of type String.

**title:** This property is of type String.

**lang:** This property is of type String; This property can raise a DOMException on setting.

**dir:** This property is of type String; This property can raise a DOMException on setting.

**className:** This property is of type String.

## AC.2.4 Object DVBHTMLFormElement

DVBHTMLFormElement has all the properties and methods of the DVBHTMLFormElement object as well as the properties and methods defined below.

The DVBHTMLFormElement object has the following properties:

**elements:** This read-only property is a DVBHTMLCollection object.

**length:** This read-only property is of type Number.

**acceptCharset:** This property is of type String.

**action:** This property is of type String.

**enctype:** This property is of type String.

**method:** This property is of type String; This property can raise a DOMException on setting.

**target:** This property is of type String.

The DVBHTMLFormElement object has the following methods:

**submit():** This method has no return value.

**reset():** This method has no return value.

See:

- [8.11.4.6.6, "DVBHTMLFormElement Interface" on page 143](#)

## AC.2.5 Object DVBHTMLSelectElement

DVBHTMLSelectElement has all the properties and methods of the DVBHTMLFormElement object as well as the properties and methods defined below.

The DVBHTMLSelectElement object has the following properties:

**type:** This read-only property is of type String.

**selectedIndex:** This property is a long object.

**value:** This property is of type String.

**length:** This read-only property is of type Number

**form:** This read-only property is a DVBHTMLFormElement object.

**options:** This read-only property is a DVBHTMLCollection object.

**disabled:** This property is of type Boolean.

**multiple:** This read-only property is of type Boolean.

**name:** This property is of type String.

**size:** This property is of type Number.

The DVBHTMLSelectElement object has the following methods:

**add(element, before):** This method has no return value.

The **element** parameter is a DVBHTMLInputElement object.

The **before** parameter is a DVBHTMLInputElement object.

This method can raise a **DOMException** object.

**remove(index)**: This method has no return value.

The **index** parameter is of type Number.

**blur()**: This method has no return value.

**focus()**: This method has no return value.

See:

- [8.11.4.6.14, "DVBHTMLSelectElement Interface" on page 150](#)

## AC.2.6 Object DVBHTMLOptionElement

DVBHTMLOptionElement has all the properties and methods of the DVBHTMLInputElement object as well as the properties and methods defined below.

The DVBHTMLOptionElement object has the following properties:

**form**: This read-only property is a DVBHTMLFormElement object.

**defaultSelected**: This property is of type Boolean.

**text**: This read-only property is of type String.

**index**: This read-only property is of type Number.

**disabled**: This property is of type Boolean.

**label**: This property is of type String.

**selected**: This property is of type Boolean.

**value**: This property is of type String.

See:

- [8.11.4.6.13, "DVBHTMLOptionElement Interface" on page 149](#)

## AC.2.7 Object DVBHTMLInputElement

DVBHTMLInputElement has all the properties and methods of the DVBHTMLInputElement object as well as the properties and methods defined below.

The DVBHTMLInputElement object has the following properties:

**defaultValue**: This property is of type String.

**defaultChecked**: This property is of type Boolean.

**form**: This read-only property is a DVBHTMLFormElement object.

**accept**: This property is of type String.

**accessKey**: This property is of type String.

**alt:** This property is of type String.

**checked:** This property is of type Boolean.

**disabled:** This property is of type Boolean.

**maxLength:** This property is of type Number.

**name:** This property is of type String.

**readOnly:** This property is of type Boolean.

**size:** This property is of type String.

**src:** This property is of type String.

**type:** This read-only property is of type String.

**useMap:** This read-only property is of type String.

**value:** This property is of type String.

The DVBHTMLInputElement object has the following methods:

**blur():** This method has no return value.

**focus():** This method has no return value.

**select():** This method has no return value.

**click():** This method has no return value.

See:

- [8.11.4.6.12, "DVBHTMLInputElement Interface" on page 147](#)

## AC.2.8 Object DVBHTMLTextAreaElement

DVBHTMLTextAreaElement has all the properties and methods of the DVBHTMLFormElement object as well as the properties and methods defined below.

The DVBHTMLTextAreaElement object has the following properties:

**defaultValue:** This property is of type String.

**form:** This read-only property is a DVBHTMLFormElement object.

**accessKey:** This property is of type String.

**cols:** This read-only property is a long object.

**disabled:** This property is of type Boolean.

**name:** This property is of type String.

**readOnly:** This property is of type Boolean.

**rows:** This property is a long object.

**type:** This read-only property is of type String.

**value:** This property is of type String.

The DVBHTMLTextAreaElement object has the following methods:

**blur():** This method has no return value.

**focus():** This method has no return value.

**select():** This method has no return value.

See:

- [8.11.4.6.15, "DVBHTMLTextAreaElement Interface" on page 151](#)

## AC.2.9 Object DVBHTMLButtonElement

DVBHTMLButtonElement has the all the properties and methods of the DVBHTMLInputElement object as well as the properties and methods defined below.

The DVBHTMLButtonElement object has the following properties:

**form:** This read-only property is a DVBHTMLFormElement object.

**accessKey:** This property is of type String.

**disabled:** This property is of type Boolean.

**name:** This property is of type String.

**type:** This read-only property is of type String.

**value:** This property is of type String.

See:

- [8.11.4.6.5, "DVBHTMLButtonElement Interface" on page 142](#)

## AC.2.10 Object DVBHTMLAnchorElement

DVBHTMLAnchorElement has the all the properties and methods of the DVBHTMLInputElement object as well as the properties and methods defined below.

The DVBHTMLAnchorElement object has the following properties:

**accessKey:** This property is of type String.

**charset:** This property is of type String.

**href:** This property is of type String.

**hreflang:** This property is of type String.

**target:** This property is of type String.

**type:** This property is of type String.

The DVBHTMLAnchorElement object has the following methods:

**blur():** This method has no return value.

**focus():** This method has no return value.

See:

- [8.11.4.6.2, "DVBHTMLAnchorElement Interface" on page 141](#)

## AC.2.11 Object DVBHTMLImageElement

DVBHTMLImageElement has all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLImageElement object has the following properties:

**lowSrc:** This property is of type String.

**alt:** This property is of type String.

**height:** This property is of type String.

**longDesc:** This property is of type String.

**src:** This property is of type String.

**useMap:** This read-only property is of type String.

**width:** This property is of type String.

See:

- [8.11.4.6.10, "DVBHTMLImageElement Interface" on page 145](#)

## AC.2.12 Object DVBHTMLObjectElement

DVBHTMLObjectElement has all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLObjectElement object has the following properties:

**form:** This read-only property is a DVBHTMLFormElement object.

**code:** This property is of type String.

**archive:** This property is of type String.

**codeBase:** This property is of type String.

**codeType:** This property is of type String.

**data:** This property is of type String.

**declare:** This property is of type Boolean.

**height:** This property is of type String.

**standby:** This property is of type String.

**tabIndex:** This property is a long object.

**type:** This property is of type String.

**useMap:** This read-only property is of type String.

**width:** This property is of type String.

**contentDocument:** This read-only property is a Document object.

See:

- [8.11.4.6.11, "DVBHTMLObjectElement Interface" on page 146](#)

## AC.2.13 Object DVBHTMLMapElement

DVBHTMLMapElement has the all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLMapElement object has the following properties:

**areas:** This read-only property is a DVBHTMLCollection object.

See:

- [8.11.4.6.3, "DVBHTMLMapElement Interface" on page 141](#)

## AC.2.14 Object DVBHTMLAreaElement

DVBHTMLAreaElement has the all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLAreaElement object has the following properties:

**accessKey:** This property is of type String.

**alt:** This property is of type String.

**href:** This property is of type String.

**noHref:** This property is of type Boolean.

**target:** This property is of type String.

See:

- [8.11.4.6.4, "DVBHTMLAreaElement Interface" on page 142](#)

## AC.2.15 Object DVBHTMLFrameSetElement

DVBHTMLFrameSetElement has the all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLFrameSetElement object has the following properties:

**cols:** This read-only property is of type String.

**rows:** This read-only property is of type String.

See:

- [8.11.4.6.8, "DVBHTMLFrameSetElement Interface" on page 145](#)

## AC.2.16 Object DVBHTMLFrameElement

DVBHTMLFrameElement has the all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLFrameElement object has the following properties:

**frameBorder:** This read-only property is of type String.

**longDesc:** This property is of type String.

**marginHeight:** This read-only property is of type String.

**marginWidth:** This read-only property is of type String.

**scrolling:** This read-only property is of type String.

**src:** This property is of type String.

**contentDocument:** This read-only property is a Document object.

See:

- [8.11.4.6.7, "DVBHTMLFrameElement Interface" on page 144](#)

## AC.2.17 Object DVBHTMLIFrameElement

DVBHTMLIFrameElement has the all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLIFrameElement object has the following properties:

**frameBorder:** This read-only property is of type String.

**longDesc:** This property is of type String.

**marginHeight:** This read-only property is of type String.

**marginWidth:** This read-only property is of type String.

**scrolling:** This read-only property is of type String.

**src:** This property is of type String.

**contentDocument:** This read-only property is a Document object.

See:

- [8.11.4.6.9, "DVBHTMLIFrameElement Interface" on page 145](#)

## AC.3 DVB-HTML event host objects

### AC.3.1 Object DVBLifecycleEvent

DVBLifecycleEvent has the all the properties and methods of the DOMEvent object as well as the properties and methods defined below.

The DVBLifecycleEvent object has the following properties:

**detail** : this readonly property is of type Number

The DVBHTMLAnchorElement object has the following methods:

**initDVBLifecycleEvent(typeArg, canBubbleArg, cancelableArg, detailArg):** This method has no return value.

The **typeArg** parameter is of type String

The **canBubbleArg** parameter is of type Boolean

The **cancelableArg** parameter is of type Boolean



The **detailArg** parameter is of type Number

See:

- [8.11.2.2.1, "Interface DVBLifecycleEvent" on page 129](#)

## AC.4 DVB-HTML environment host objects

### AC.4.1 Navigator Object

**appCodeName**: this read-only property is of type String ;

**appName**: this read-only property is of type String ;

**appVersion**: this read-only property is of type String ;

**userAgent**: this read-only property is of type String ;

See:

- [8.11.7.2.1, "Navigator Object" on page 154](#)

### AC.4.2 Window Object

Window object has the all the properties and methods defined below.

The Window object has the following properties:

**document**: this read-only property is of type DVBHTMLDocument ;

**DOMImplementation**: this read-only property is of type DVBDOMImplementation ;

**frames**: this read-only property is of type DVBHTMLCollection ;

**length**: this read-only property is of type Number

**location**: this property is of type Location ;

**name**: this read-only property is of type String ;

**navigator**: this read-only property is of type Navigator

**parent**: this read-only property is of type Window ;

**window**: this read-only property is of type Window

**self**: this read-only property is of type Window ;

**status**: this property is of type String ;

**defaultStatus**: this property is of type String ;

**top**: this read-only property is of type Window

The Window object has the following methods:

**clearTimeout(timerId)**: This method has no return value

The parameter **timerId** is of type Number

**open(url, name, features)**: This method returns a Window object

The parameter **url** is of type String

The parameter **name** is of type String

The parameter **features** is of type String

**setTimeout(statement, delay)**: This method returns a Number

The **statement** parameter is of type String

The **delay** parameter is of type Number

See:

- [8.11.7.2.2, "Window object" on page 154](#)

### AC.4.3 Location object

:Location has the all the properties and methods defined below.

The Location object has the following properties:

**hash**; This property is of type String

**host**; This property is of type String

**hostname**; This property is of type String

**href**; This property is of type String

**pathname**; This property is of type String

**port**; This property is of type String

**protocol**; This property is of type String

**search**; This property is of type String

See:

- [8.11.7.2.3, "Location object." on page 157](#)

## AC.5 DVB-HTML CSS host objects

### AC.5.1 DVBCSSStyle

The DVBElement object for elements supporting the 'style' attribute has the additional properties:

**style**: This read-only property is of type CSS2Properties

See:

- [8.11.8.1.1, "DVBCSSInlineStyle" on page 157](#)

### AC.5.2 DVBCSSStyle

The DVBElement object for the root element in a top level document has the additional properties:

**viewport**: This read-only property is of type DVBCSSViewportRule

See:

- [8.11.8.1.2, "DVBCSSStyle" on page 158](#)

### AC.5.3 DVBCSSViewportRule

DVBCSSviewportRule has the all the properties and methods of the CSSRule object as well as the properties and methods dened below.

**style:** This read-only property is of type DVBCSSViewportProperties

See:

- [8.11.8.1.3, "DVBCSSViewportRule" on page 158](#)

### AC.5.4 DVBCSSViewportProperties

DVBCSSViewportProperties has the following properties:

**pseudoClass:** This property is of type String

**area:** This property is of type String

**backgroundVideoTransform:** This property is of type Array of String

**backgroundVideo:** This property is of type Array of String

**backgroundVideoPreserveAspect:** This property is of type Array of String

**backgroundVideoClip:** This property is of type Array of String

**backgroundVideoRectangle:** This property is of type Array of String

**background:** This property is of type String

**backgroundImageRectangle:** This property is of type String

**initial:** This property is of type String

**verticalResolution:** This property is of type Number

**horizontalResolution:** This property is of type Number

**scene:** This property is of type String"

See:

- [8.11.8.1.4, "DVBCSSViewportProperties" on page 158](#)

---

## Annex AD (normative): Support for DVB-HTML

### AD.1 Java bindings to DVB extensions

#### AD.1.1 The org.dvb.dom.dvbhtml package

All of these are in the package `org.dvb.dom.dvbhtml`.

##### AD.1.1.1 DVBHTMLButtonElement

```
public interface DVBHTMLButtonElement
  extends DVBHTMLFormElement {
  public String getAccessKey();
  public void setAccessKey(String key);
  public boolean getDisabled();
  public void setDisabled(boolean disabled);
  public DVBHTMLFormElement getForm();
  public String getName();
  public void setName(String name);
  public String getType();
  public void setValue(String value);
  public String getValue();
  public void blur();
  public void focus();
};
```

##### AD.1.1.2 DVBHTMLCollection

See:

- [8.11.4.5.1, "DVB-HTMLCollection Interface" on page 137](#)
- [AC.2.1, "Object DVBHTMLCollection" on page 1284](#)

```
public interface DVBHTMLCollection {
//Methods
  public Node getIdentifiedItem(String name)
  public Node getItem(Number index)
  public Number getLength()
}
```

##### AD.1.1.3 DVBHTMLDocument

See:

- [8.11.4.5.2, "DVBHTMLDocument Interface" on page 138](#)
- [AC.2.2, "Object DVBHTMLDocument" on page 1284](#)

```
public interface DVBHTMLDocument
  extends Document {
//Methods
  public DVBHTMLCollection getAnchors()
  public DVBHTMLFormElement getBody()
  public String getCookie()
  public String getDomain()
  public DVBHTMLCollection getForms()
  public DVBHTMLCollection getImages()
  public DVBHTMLCollection getLinks()
  public String getReference()
  public String getTitle()
  public void setTitle(String title)
  public String getURL()
  public void open()
}
```

```

        throws IllegalStateException
    public void close()
        throws IllegalStateException
    public void write(String text)
        throws IllegalStateException
    public void writeIn(String text)
        throws IllegalStateException
}

```

#### AD.1.1.4 DVBHTMLElement

See:

- [8.11.4.6.1, "DVBHTMLElement Interface" on page 140](#)
- [AC.2.3, "Object DVBHTMLElement" on page 1285](#)

```

public interface DVBHTMLElement
    extends Element {
    public String getClassName()
    public void setClassName()
    public String getDir()
    public void setDir()
    public String getId()
    public void setId(String id)
    public String getLang()
    public void setLang(String lang)
    public String getTitle()
    public void setTitle(String)
}

```

#### AD.1.1.5 DVBHTMLFormElement

See:

- [8.11.4.6.6, "DVBHTMLFormElement Interface" on page 143](#)
- [AC.2.4, "Object DVBHTMLFormElement" on page 1286](#)

```

public interface DVBHTMLFormElement
    extends DVBHTMLElement {
    public String getAction()
    public void setAction(String action)
    public String getCharSet()
    public void setCharSet(String charset)
    public DVBHTMLCollection getElements()
    public String getEncType()
    public void setEncType(String action)
    public long getLength()
    public String getMethod()
    public void setMethod(String action)
    public String getTarget()
    public void setTarget(String action)
    public void submit()
    public void reset()
}

```

#### AD.1.1.6 DVBHTMLSelectElement

See:

- [8.11.4.6.14, "DVBHTMLSelectElement Interface" on page 150](#)
- [AC.2.5, "Object DVBHTMLSelectElement" on page 1286](#)

```

public interface DVBHTMLSelectElement
    extends DVBHTMLElement {
    public String getType()
}

```

```

public long getSelectedIndex()
public void setSelectedIndex(long index)
public String getValue()
public void setValue(String value)
public long getLength()
public DVBHTMLFormElement getForm()
public DVBHTMLCollection getOptions()
public boolean isDisabled()
public void setDisabled(Boolean disabled)
public boolean isMultiple()
public String getName()
public void setName(String name)
public long getSize()
public void setSize(long index)
public long getTabIndex()
public void setTabIndex(long index)
public void add(DVBHTMLFormElement element, DVBHTMLFormElement before)
    throws DOMException
public void remove(long index)
public void blur()
public void focus()
}

```

### AD.1.1.7 DVBHTMLOptionElement

See:

- [8.11.4.6.13, "DVBHTMLOptionElement Interface" on page 149](#)
- [AC.2.6, "Object DVBHTMLOptionElement" on page 1287](#)

```

public interface DVBHTMLOptionElement
    extends DVBHTMLFormElement {
    public boolean isDefaultSelected()
    public void setDefaultSelected(boolean defaultselected)
    public boolean isSelected()
    public void setSelected(boolean selected)
    public boolean isDisabled()
    public void setDisabled(boolean disabled)
    public DVBHTMLFormElement getForm()
    public long getIndex()
    public String getLabel()
    public void setLabel(String label)
    public String getText()
    public void setText(String text)
    public String getValue()
    public void setValue(String value)
}

```

### AD.1.1.8 DVBHTMLInputElement

See:

- [8.11.4.6.12, "DVBHTMLInputElement Interface" on page 147](#)
- [AC.2.7, "Object DVBHTMLInputElement" on page 1287](#)

```

public interface DVBHTMLInputElement
    extends DVBHTMLFormElement {
    public boolean isChecked()
    public void setChecked(boolean checked)
    public boolean isDefaultChecked()
    public void setDefaultChecked(boolean defaultchecked)
    public boolean isDisabled()
    public void setDisabled(boolean disabled)
    public boolean isReadOnly()
    public void setReadOnly(boolean readonly)
    public String getAccept()
    public void setAccept(String accept)
}

```

```

public String getAccessKey()
public void setAccessKey(String accesskey)
public String getAlt()
public void setAlt(String alt)
public String getDefaultValue()
public void setDefaultValue(String value)
public DVHTMLFormElement getForm()
public long getMaxLength()
public void setMaxLength(long length)
public String getName()
public void setName(String name)
public String getSize()
public void setSize(String size)
public String getSrc()
public void setSrc(String src)
public String getType()
public void setType(String type)
public String getUseMap()
public String getValue()
public void setValue(String value)
public void blur()
public void click()
public void focus()
public void select()
}

```

### AD.1.1.9 DVHTMLTextAreaElement

See:

- [8.11.4.6.15, "DVHTMLTextAreaElement Interface" on page 151](#)
- [AC.2.8, "Object DVHTMLTextAreaElement" on page 1288](#)

```

public interface DVHTMLTextAreaElement
extends DVHTMLFormElement {
public boolean isDisabled()
public void setDisabled(boolean disabled)
public String getAccessKey()
public void setAccessKey(String accesskey)
public String getDefaultValue()
public void setDefaultValue(String value)
public DVHTMLFormElement getForm()
public String getName()
public void setName(String name)
public String getType()
public void setType(String type)
public String getValue()
public void setValue(String value)
}

```

### AD.1.1.10 DVHTMLAnchorElement

See:

- [8.11.4.6.2, "DVHTMLAnchorElement Interface" on page 141](#)
- [AC.2.10, "Object DVHTMLAnchorElement" on page 1289](#)

```

public interface DVHTMLAnchorElement
extends DVHTMLFormElement {
public String getAccessKey()
public void setAccessKey(String accesskey)
public String getCharSet()
public void setCharSet(String charset)
public String getHref()
public void setHref(String href)
public String getHrefLang()
public void setHrefLang(String hreflang)
}

```

```

    public String getTarget()
    public void setTarget(String target)
    public String getType()
    public void setType(String type)
    public void blur()
    public void focus()
}

```

### AD.1.1.11 DVBHTMLImageElement

See:

- [8.11.4.6.10, "DVBHTMLImageElement Interface" on page 145](#)
- [AC.2.11, "Object DVBHTMLImageElement" on page 1290](#)

```

public interface DVBHTMLImageElement
    extends DVBHTMLMElement {
    public String getLowSrc()
    public void setLowSrc(String lowsrc)
    public String getAlt()
    public void setAlt(String alt)
    public String getHeight()
    public void setHeight(String height)
    public String getLongDesc()
    public void setLongDesc(String longdesc)
    public String getArc()
    public void setArc(String arc)
    public String getUsemap()
    public void setUseMap(String usemap)
    public String getWidth()
    public void setWidth(String width)
}

```

### AD.1.1.12 DVBHTMLObjectElement

See:

- [8.11.4.6.11, "DVBHTMLObjectElement Interface" on page 146](#)
- [AC.2.12, "Object DVBHTMLObjectElement" on page 1290](#)

```

public interface DVBHTMLObjectElement
    extends DVBHTMLMElement {
    public boolean isDeclare()
    public void setDeclare(boolean declare)
    public DVBHTMLFormElement getForm()
    public String getCode()
    public void setCode(String code)
    public String getArchive()
    public void setArchive(String archive)
    public String getCodeBase()
    public void setCodeBase(String codebase)
    public String getCodeType()
    public void setCodeType(String codetype)
    public String getData()
    public void setData(String data)
    public String getHeight()
    public void setHeight(String height)
    public String getStandby()
    public void setStandby(String standby)
    public long getTabIndex()
    public void setTabIndex(String index)
    public String getType()
    public void setType(String type)
    public String getUseMap()
    public void setUseMap(String usemap)
    public String getWidth()
    public void setWidth(String width)
}

```



```

    public Document getContentDocument()
}

```

### AD.1.1.13 DVBHTMLMapElement

See:

- [8.11.4.6.3, "DVBHTMLMapElement Interface" on page 141](#)
- [AC.2.13, "Object DVBHTMLMapElement" on page 1291](#)

```

public interface DVBHTMLMapElement
    extends DVBHTMLMElement {
    public DVBHTMLCollection getAreas()
}

```

### AD.1.1.14 DVBHTMLAreaElement

See:

- [8.11.4.6.4, "DVBHTMLAreaElement Interface" on page 142](#)
- [AC.2.14, "Object DVBHTMLAreaElement" on page 1291](#)

```

public interface DVBHTMLAreaElement
    extends DVBHTMLMElement {
    public boolean isNoHRef()
    public void setNoHRef(boolean nohref)
    public String getAccessKey()
    public void setAccessKey(String accesskey)
    public String getAlt()
    public void setAlt(String alt)
    public String getHRef()
    public void setHRef(String href)
    public String getTarget()
    public void setTarget(String target)
}

```

### AD.1.1.15 DVBHTMLFrameSetElement

See:

- [8.11.4.6.8, "DVBHTMLFrameSetElement Interface" on page 145](#)
- [AC.2.15, "Object DVBHTMLFrameSetElement" on page 1291](#)

```

public interface DVBHTMLFrameSetElement
    extends DVBHTMLMElement {
    public String getCols()
    public void setCols(String cols)
    public String getRows()
    public void setRows(String rows)
}

```

### AD.1.1.16 DVBHTMLFrameElement

See:

- [8.11.4.6.7, "DVBHTMLFrameElement Interface" on page 144](#)
- [AC.2.16, "Object DVBHTMLFrameElement" on page 1291](#)

```

public interface DVBHTMLFrameElement
    extends DVBHTMLMElement {
    public Document getContentDocument()
    public String getFrameBorder()
}

```

```

public void setFrameBorder(String border)
public String getLongDesc()
public void setLongDesc(String longdesc)
public String getMarginHeight()
public void setMarginHeight(String marginheight)
public String getMarginWidth()
public void setMarginWidth(String marginwidth)
public String getScrolling()
public void setScrolling(String scrolling)
public String getSrc()
public void setSrc(String src)
}

```

### AD.1.1.17 DVBHTMLIFrameElement

See:

- 8.11.4.6.9, "DVBHTMLIFrameElement Interface" on page 145
- AC.2.17, "Object DVBHTMLIFrameElement" on page 1292

```

public interface DVBHTMLIFrameElement
extends DVHTMLElement {
public String getAlign()
public void setAlign(String align)
public Document getContentDocument()
public String setFrameBorder()
public void setFrameBorder(String border)
public String getHeight()
public void setHeight(String height)
public String getLongDesc()
public void setLongDesc(String longdesc)
public String getMarginHeight()
public void setMarginHeight(String marginheight)
public String getMarginWidth()
public void setMarginWidth(String marginwidth)
public String getName()
public void setName(String name)
public String getScrolling()
public void setScrolling(String scrolling)
public String getSrc()
public void setSrc(String src)
public String getWidth()
public void setWidth(String width)
}

```

## AD.1.2 The org.dvb.dom.css package

All of these are in the package `org.dvb.dom.css`.

### AD.1.2.1 DVBCSSInlineStyle

This interface shall be implemented by objects that implement the `org.w3c.dom.Element` interface and which represent elements that support the style attribute.

```

public interface DVBCSSInlineStyle {
public CSS2Properties getStyle();
};

```

See:

- 8.11.8.1.1, "DVBCSSInlineStyle" on page 157
- AC.5.1, "DVBCSSInlineStyle" on page 1294

### AD.1.2.2 DVBCSSStyle

This interface shall be implemented by objects that implement the `org.w3c.dom.Element` interface that represent the root element of a document.

```
public interface DVBCSSStyle {
    public DVBCSSViewportRule getViewport();
};
```

See:

- [8.11.8.1.2, "DVBCSSStyle" on page 158](#)
- [AC.5.2, "DVBCSSStyle" on page 1294](#)

### AD.1.2.3 DVBCSSViewportRule

```
public interface DVBCSSViewportRule extends org.w3c.dom.CSSRule {
    public DVBCSSViewportProperties getStyle();
};
```

See:

- [8.11.8.1.3, "DVBCSSViewportRule" on page 158](#)
- [AC.5.3, "DVBCSSViewportRule" on page 1295](#)

### AD.1.2.4 DVBCSSViewportProperties

```
public interface DVBCSSViewportProperties {
    public String getPseudoClass();
    public String getArea();
    public void setArea(String area);
    public String[] getBackgroundVideoTransform();
    public void setBackgroundVideoTransform(String transform[]);
    public String[] getBackgroundVideo();
    public void setBackgroundVideo(String video[]);
    public String[] getBackgroundVideoPreserveAspect();
    public void setBackgroundVideoPreserveAspect(String aspect[]);
    public String[] getBackgroundVideoClip();
    public void setBackgroundVideoClip(String clip[]);
    public String[] getBackgroundVideoRectangle();
    public void setBackgroundVideoRectangle(String rectangle[]);
    public String getBackground();
    public void setBackground(String background);
    public String getBackgroundImageRectangle();
    public void setBackgroundImageRectangle(String rect);
    public String getInitial();
    public void setInitial(String initial);
    public int getVerticalResolution();
    public void setVerticalResolution(int res);
    public int getHorizontalResolution();
    public void setHorizontalResolution(int res);
    public String getScene();
    public void setScene(String scene);
};
```

See:

- [8.11.8.1.4, "DVBCSSViewportProperties" on page 158](#)
- [AC.5.4, "DVBCSSViewportProperties" on page 1295](#)

## AD.1.3 The org.dvb.dom.environment package

All of these are in the package org.dvb.dom.environment.

### AD.1.3.1 Navigator

See:

- [8.11.7.2.1, "Navigator Object" on page 154](#)
- [AC.4.1, "Navigator Object" on page 1293](#)

```
public class Navigator
  extends java.lang.Object {
  public Navigator()
  public Navigator(String appname, String appcodename,
    String appversion, String useragent)
  public String getAppCodeName()
  public String getAppName()
  public String getAppVersion()
  public String getUserAgent()
}
```

### AD.1.3.2 Window

See:

- [8.11.7.2.2, "Window object" on page 154](#)
- [AC.4.2, "Window Object" on page 1293](#)

```
public class Window
  extends java.lang.Object {
  public Window()
  public Window(DVBHTMLDocument document, DVBDOMImpl impl,
    DVBHTMLCollection frames, long length, Navigator navigator,
    Window parent, Window window, Window top)
  public String getDefaultStatus()
  public void setDefaultStatus(String defaultstatus)
  public DVBHTMLDocument getDocument()
  public DVBDOMImpl getDVBDOMImpl()
  public DVBHTMLCollection getFrames()
  public long getLength()
  public Location getLocation()
  public void setLocation(Location location)
  public String getName()
  public void setName(String name)
  public Navigator getNavigator()
  public Window getParent()
  public Window getWindow()
  public Window getSelf()
  public String getStatus()
  public void setStatus(String status)
  public Window getTop()
  public void open(String url, String name, String features)
}
```

### AD.1.3.3 Location

See:

- [8.11.7.2.3, "Location object." on page 157](#)
- [AC.4.3, "Location object" on page 1294](#)

```
public class Location
  extends java.lang.Object {
  public Location()
```

```
public String getNash()
public void setHash(String hash)
public String getHost()
public void setHost(String host)
public String getHostName()
public void setHostName(String hostname)
public String getHRef()
public void setHRef(String href)
public String getPathName()
public void setPathName(String pathname)
public String getPort()
public void setPort(String port)
public String getProtocol()
public void setProtocol(String protocol)
public String getSearch()
public void setSearch(String search)
}
```

## AD.1.4 The org.dvb.dom.event package

All of these are in the package org.dvb.dom.event.

### AD.1.4.1 DVBLifecycleEvent

See:

- [8.11.2.2.1, "Interface DVBLifecycleEvent" on page 129](#)
- [AC.3.1, "Object DVBLifecycleEvent" on page 1292](#)

```
public class DVBLifecycleEvent
extends java.util.event {
public DVBLifecycleEvent()
public DVBLifecycleEvent(String typearg, boolean canbubblearg,
boolean candisablearg, long detailarg)
public long getDetail()
}
```

# Package

# org.dvb.dom.bootstrap

## Description

Provides the entry point to the W3C DOM APIs for Java applications.

## Class Summary

### Interfaces

<code>DocumentAction</code>	DocumentAction is used for an action that modifies a W3C Document.
<code>DocumentFactory</code>	DocumentFactory contains bootstrap methods for applications embedded in a document to access the document object model of the document in which they are contained.
<code>MultipleDocumentsAction</code>	MultipleDocumentAction is used for an action that modifies zero or more W3C Document instances.

# org.dvb.dom.bootstrap DocumentAction

## Syntax

```
public interface DocumentAction
```

## Description

DocumentAction is used for an action that modifies a W3C Document. When an application wishes to access a Document, it creates an instance of a class that implements DocumentAction, and passes this instance to the system. The system then calls back to the user code via the DocumentAction interface.

## See Also:

`MultipleDocumentsAction`, `DocumentFactory`

## Methods

---

### **run(Document)**

```
public void run(org.w3c.dom.Document doc)
```

Access and/or modify a W3C DOM Document. All modifications must take place during this callback. The results of retaining and using DOM references outside the context of this callback are undefined.

#### **Parameters:**

`doc` - the DOM document to modify

# org.dvb.dom.bootstrap DocumentFactory

## Syntax

```
public interface DocumentFactory
```

## Description

DocumentFactory contains bootstrap methods for applications embedded in a document to access the document object model of the document in which they are contained.

## Since:

MHP 1.1

## Fields

---

### DOM

```
public static final java.lang.String DOM
```

The property string to use with `XletContext.getXletProperty` in order to obtain the DocumentFactory for this Xlet (if one exists).

## Methods

---

### performAction(DocumentAction)

```
public void performAction(DocumentAction act)
```

Perform an action on the document that contains the Xlet controlled by the given XletContext.

#### Parameters:

`act` - The action to perform. It will be called by the system either synchronously, or on a system thread.

---

### performActionOnFrames(String[], MultipleDocumentsAction)

```
public void performActionOnFrames(java.lang.String[] names, MultipleDocumentsAction act)
```

Perform an action on a set of documents, each contained in a frame that is a part of the same application as the Xlet controlled by the given XletContext.

#### Parameters:

`names` - The names of the desired frames.

`act` - The action to perform. It will be called by the system either synchronously, or on a system thread.



---

**performActionReadOnly(DocumentAction)**

```
public void performActionReadOnly(DocumentAction act)
```

Perform an action on the document that contains the Xlet controlled by the given XletContext. The action is called without any ability to modify the DOM.

**Parameters:**

`act` - The action to perform. It will be called by the system either synchronously, or on a system thread. Attempts by this action to modify the DOM shall fail.

# org.dvb.dom.bootstrap MultipleDocumentsAction

## Syntax

```
public interface MultipleDocumentsAction
```

## Description

MultipleDocumentAction is used for an action that modifies zero or more W3C Document instances. When an application wishes to access a number of Documents simultaneously, it creates an instance of a class that implements MultipleDocumentAction, and passes this instance to the system. The system then calls back to the user code via the MultipleDocumentAction interface.

## See Also:

[DocumentAction](#), [DocumentFactory](#)

## Methods

---

### run(Document[])

```
public void run(org.w3c.dom.Document[] docs)
```

Access and/or modify a set of W3C DOM Documents. All modifications must take place during this callback. The results of retaining and using DOM references outside the context of this callback are undefined.

#### Parameters:

`docs` - The array of documents on which to operate. If a requested document is not found, the corresponding entry in this array will be null.

# Package

# org.dvb.dom.inner

## Description

Provides support for embedding DVB-HTML applications within the user interface of a DVB-J application.

## Class Summary

### Classes

<code>HTMLApplication</code>	Encapsulates the information needed to define a DVB-HTML application.
<code>HTMLInnerApplication- Container</code>	Represents embedding of an DVB-HTML inner application within the user interface of a DVB-J application.

# org.dvb.dom.inner HTMLApplication

## Syntax

```
public class HTMLApplication extends InnerApplication
```

```
java.lang.Object
|
+--InnerApplication
|
+--org.dvb.dom.inner.HTMLApplication
```

## Description

Encapsulates the information needed to define a DVB-HTML application.

## Constructors

---

### HTMLApplication(URL, String, String)

```
public HTMLApplication(java.net.URL physicalRoot, java.lang.String initialPathBytes,
    java.lang.String parameters)
```

Constructor for instances of the class. The semantics of these shall be as defined in the main body of this specification for the equivalent information when provided through an AIT.

#### Parameters:

`physicalRoot` - the physical root of the application entry point.

`initialPathBytes` - a string specifying the URL path component to the entry point document. This path is relative to the root defined in the `physicalRoots` parameter.

`parameters` - the string that is appended to the application initial path as parameters

---

### HTMLApplication(URL, String, String, String[], String[])

```
public HTMLApplication(java.net.URL physicalRoot, java.lang.String initialPathBytes,
    java.lang.String parameters, java.lang.String[] label,
    java.lang.String[] regex)
```

Constructor for instances of the class. The semantics of these shall be as defined in the main body of this specification for the equivalent information when provided through an AIT.

#### Parameters:

`physicalRoot` - the physical root of the application entry point.

`initialPathBytes` - a string specifying the URL path component to the entry point document. This path is relative to the root defined in the `physicalRoots` parameter.

`parameters` - the string that is appended to the application initial path as parameters

`label` - an array of one or more strings specifying a label that is associated with the set of data identified by the regular expression. This label can be used for pre-fetching in a transport specific manner.

`regex` - an array of one or more strings specifying a regular expressions that can generate all URLs that are in the domain of the application.

#### Throws:

`IllegalArgumentException` - if either the `regex` or `label` parameters are an array of length zero or if the lengths of these two arrays are not the same

# org.dvb.dom.inner HTMLInnerApplicationContainer

## Syntax

public class HTMLInnerApplicationContainer extends [InnerApplicationContainer](#)

```

java.lang.Object
|
+--java.awt.Component
    |
    +--org.havi.ui.HComponent
        |
        +--InnerApplicationContainer
            |
            +--org.dvb.dom.inner.HTMLInnerApplicationContainer
  
```

## All Implemented Interfaces:

org.havi.ui.HMatteLayer, org.havi.ui.HNavigable, org.havi.ui.HNavigationInputPreferred, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, org.dvb.ui.TestOpacity

## Description

Represents embedding of an DVB-HTML inner application within the user interface of a DVB-J application.

## Constructors

---

### HTMLInnerApplicationContainer(HTMLApplication)

```
public HTMLInnerApplicationContainer(HTMLApplication a)
```

Construct an instance of this class with a DVB-HTML application as its content. The instance is initialised to present the entry point document of the application.

#### Parameters:

a - the DVB-HTML application

#### Throws:

[IOException](#) - if an error occurred while reading the code or data for the inner application

## Methods

---

### performAction(DocumentAction)

```
public void performAction(DocumentAction act)
```

Perform an action on the DVB-HTML document.

#### Parameters:

act - The action to perform. It will be called by the system either synchronously, or on a system thread.

---

## **Annex AĒ (normative): Inner Applications**

# Package

# org.dvb.application.inner

## Description

Provides support for embedding other interactive applications within the user interface of a DVB-J application.

### Class Summary

#### Interfaces

<code>DVBScene</code>	Represents various features which are common to the top level Container of applications whether running stand-alone or embedded in a DVB-HTML application.
<code>InnerApplicationListener</code>	Listener for events related to inner applications

#### Classes

<code>InnerApplication</code>	Encapsulates the information needed to define an inner application.
<code>InnerApplicationContainer</code>	Represents embedding of an inner application within the user interface of a DVB-J application.
<code>InnerApplicationEvent</code>	<code>InnerApplicationEvent</code> is used to notify applications which have registered interest in events coming from inner applications that such an event has happened.

# org.dvb.application.inner DVBScene

## Syntax

```
public interface DVBScene
```

## Description

Represents various features which are common to the top level Container of applications whether running stand-alone or embedded in a DVB-HTML application. All integer constants used in this class are those defined in the class `org.havi.ui.HScene`.

## Methods

---

### addWindowListener(WindowListener)

```
public void addWindowListener(java.awt.event.WindowListener wl)
```

Add a listener to receive any `java.awt.event.WindowEvents` sent from this `DVBScene`. If the listener has already been added further calls will add further references to the listener, which will then receive multiple copies of a single event.

#### Parameters:

`wl` - The `java.awt.event.WindowListener` to be notified of any `java.awt.event.WindowEvents`.

---

### enableShortcuts(boolean)

```
public void enableShortcuts(boolean enable)
```

Enables or disables all short cuts that are currently set on the Scene. To enable or disable a single shortcut use `org.dvb.application.inner.DVBScene#addShortcut`, `addShortcut` or `removeShortcut(int)`.

Note `enableShortcuts(boolean)` does not remove existing added `DVBScene` shortcuts - they are merely disabled and may be subsequently re-enabled with `enableShortcuts(boolean)`.

#### Parameters:

`enable` - a value of true indicates all shortcuts are to be enabled, and a value of false indicates all shortcuts are to be disabled.

---

### getAllShortcutKeycodes()

```
public int[] getAllShortcutKeycodes()
```

Returns all keycodes added in the `DVBScene` as shortcuts.

#### Returns:

all keycodes added in the `DVBScene` as shortcuts, there are no ordering guarantees.

---

### getBackgroundImage()

```
public java.awt.Image getBackgroundImage()
```

Retrieve any image used as a background for this `DVBScene`.



**Returns:**

an image used as a background, or `null` if no image is set. Note that depending on the current render mode any image set may not actually be rendered.

**See Also:**

[setRenderMode\(int\)](#)

---

**getBackgroundMode()**

```
public int getBackgroundMode()
```

Get the background mode of this [DVBScene](#). The return value specifies whether the paint method should draw the background (i.e. a rectangle filling the bounds of the [DVBScene](#)).

**Returns:**

one of `NO_BACKGROUND_FILL` or `BACKGROUND_FILL`.

---

**getFocusOwner()**

```
public java.awt.Component getFocusOwner()
```

Returns the child component of this [DVBScene](#) which has focus if and only if this [DVBScene](#) is active.

**Returns:**

the component with focus, or `null` if no children have focus assigned to them.

---

**getRenderMode()**

```
public int getRenderMode()
```

Get the rendering mode of any background image associated with this [DVBScene](#).

**Returns:**

the rendering mode, one of `IMAGE_NONE`, `IMAGE_STRETCH`, `IMAGE_CENTER` or `IMAGE_TILE`.

---

**getShortcutComponent(int)**

```
public org.havi.ui.HActionable getShortcutComponent(int keyCode)
```

Retrieve the `HActionable` associated with the specified shortcut key.

**Parameters:**

`keyCode` - the shortcut key code to be queried for an associated `HActionable`.

**Returns:**

the `HActionable` associated with the specified key if `keyCode` is a valid shortcut key for this [DVBScene](#), `null` otherwise.

---

**getShortcutKeycode(HActionable)**

```
public int getShortcutKeycode(org.havi.ui.HActionable comp)
```

Returns the keycode associated with the specified `HActionable` component.

**Parameters:**

`comp` - the `HActionable` to return the keycode that it is associated with.

**Returns:**

the keycode associated with the specified `HActionable` component, if it is currently a valid shortcut "target", otherwise return `java.awt.event.KeyEvent#VK_UNDEFINED`.

---

---

**isDoubleBuffered()**

```
public boolean isDoubleBuffered()
```

Returns `true` if all the drawing done during the update and paint methods for this specific `DVBScene` object is automatically double buffered.

**Returns:**

`true` if all the drawing done during the update and paint methods for this specific `DVBScene` object is automatically double buffered, or `false` if drawing is not double buffered. The default value for the double buffering setting is platform-specific.

---

**isEnabledShortcuts()**

```
public boolean isEnabledShortcuts()
```

Returns the status of all short cuts that are currently set on the `DVBScene`.

**Returns:**

`true` if shortcuts are enabled, `false` otherwise.

**See Also:**

`enableShortcuts(boolean)`

---

**isOpaque()**

```
public boolean isOpaque()
```

Returns `true` if the entire `DVBScene` area, as given by the `java.awt.Component#getBounds` method, is fully opaque, i.e. its paint method (or surrogate methods) guarantee that all pixels are painted in an opaque `Color`.

By default, the return value depends on the value of the current background mode, as set by the `setBackgroundMode(int)` method. The return value should be overridden by subclasses that can guarantee full opacity. The consequences of an invalid overridden value are implementation specific.

**Returns:**

`true` if all the pixels within the area given by the `java.awt.Component#getBounds` method are fully opaque, i.e. its paint method (or surrogate methods) guarantee that all pixels are painted in an opaque `Color`, otherwise `false`.

---

**isVisible()**

```
public boolean isVisible()
```

Determines if the `DVBScene` (or more properly its added child components) is Visible. Initially an `DVBScene` is invisible.

**Returns:**

`true` if the `DVBScene` is visible; `false` otherwise.

---

**removeShortcut(int)**

```
public void removeShortcut(int keyCode)
```

Removes the specified short-cut key. if the specified short-cut key is not registered, the method has no effect

**Parameters:**

`keyCode` - The keycode that represents the short cut

---

**removeWindowListener(WindowListener)**

```
public void removeWindowListener(java.awt.event.WindowListener wl)
```

Remove a listener so that it no longer receives any `java.awt.event.WindowEvents`. If the specified listener is not registered, the method has no effect. If multiple references to a single listener have been registered it should be noted that this method will only remove one reference per call.

**Parameters:**

`wl` - The `java.awt.event.WindowListener` to be removed from notification of any `java.awt.event.WindowEvents`.

---

**setBackgroundImage(Image)**

```
public void setBackgroundImage(java.awt.Image image)
```

Set an image which shall be painted in the background of the `DVBScene`, after the background has been drawn according to the current mode set with `setBackgroundMode(int)`, but before any children are drawn. The image is rendered according to the current render mode set with `setRenderMode(int)`.

Note that the use of a background image in this way may affect the return value of the `isOpaque()` method, depending on the image and the current rendering mode.

**Parameters:**

`image` - the image to be used as a background. If this parameter is `null` any current image is removed. Note that depending on the current render mode any image set may not actually be rendered.

**See Also:**

`setRenderMode(int)`

---

**setBackgroundMode(int)**

```
public void setBackgroundMode(int mode)
```

Set the background mode of this `DVBScene`. The value specifies whether the paint method should draw the background (i.e. a rectangle filling the bounds of the `DVBScene`).

Note that the background mode will affect the return value of the `isOpaque()` method, depending on the value of the `mode` parameter. A fill mode of `BACKGROUND_FILL` implies that `isOpaque()` must return `true`.

**Parameters:**

`mode` - one of `NO_BACKGROUND_FILL` or `BACKGROUND_FILL`. If `mode` is not a valid value, an `IllegalArgumentException` will be thrown.

---

**setRenderMode(int)**

```
public boolean setRenderMode(int mode)
```

Set the rendering mode of any background image associated with this `DVBScene`.

Note that the minimum requirement is to support only the `IMAGE_NONE` mode. Support of the other modes is platform and implementation specific.

**Parameters:**

`mode` - the rendering mode, one of `IMAGE_NONE`, `IMAGE_STRETCH`, `IMAGE_CENTER` or `IMAGE_TILE`.

**Returns:**

`true` if the mode was set successfully, `false` if the mode is not supported by the platform.

# org.dvb.application.inner InnerApplication

## Syntax

```
public abstract class InnerApplication
```

```
java.lang.Object
```

```
|
```

```
+--org.dvb.application.inner.InnerApplication
```

## Direct Known Subclasses:

```
HTMLApplication
```

## Description

Encapsulates the information needed to define an inner application.

## Constructors

---

### InnerApplication()

```
protected InnerApplication()
```

A protected constructor. This shall not be used by inter-operable applications.

# org.dvb.application.inner InnerApplicationContainer

## Syntax

```
public abstract class InnerApplicationContainer extends org.havi.ui.HComponent implements
    org.havi.ui.HNavigable
```

```
java.lang.Object
|
+--java.awt.Component
    |
    +--org.havi.ui.HComponent
        |
        +--org.dvb.application.inner.InnerApplicationContainer
```

## Direct Known Subclasses:

[HTMLInnerApplicationContainer](#)

## All Implemented Interfaces:

org.havi.ui.HMatteLayer, org.havi.ui.HNavigable, org.havi.ui.HNavigationInputPreferred, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, org.dvb.ui.TestOpacity

## Description

Represents embedding of an inner application within the user interface of a DVB-J application. The inner application shall be started when the object of this class is constructed. The behaviour of instances of this class when an inner application exits is either implementation or inner application format dependent.

When an instance of this class gains input focus, shall gain the input focus. The meaning of this is specific to the format of the inner application concerned. Inner applications are allowed to consume the events VK\_UP, VK\_DOWN, VK\_LEFT and VK\_RIGHT. While this is happening, the HNavigable focus management shall be disabled and the actions defined by `setFocusTraversal` shall not happen. The mechanisms by which inner applications start and stop consuming these events are specific to the content format of the inner application.

## Constructors

---

### InnerApplicationContainer(InnerApplication)

```
protected InnerApplicationContainer(InnerApplication a)
```

Construct an instance of this class with a particular inner application as its content. This shall not be used by inter-operable applications.

#### Parameters:

a - the inner application

#### Throws:

IOException - if an error occurred while reading the code or data for the inner application

## Methods

---

**addInnerApplicationListener(InnerApplicationListener)**

```
public void addInnerApplicationListener(InnerApplicationListener l)
```

Add a listener for events when the inner application exits.

**Parameters:**

`l` - the listener to be notified when the inner application exits

---

**getGainFocusSound()**

```
public org.havi.ui.HSound getGainFocusSound()
```

Get the sound associated with the gain focus event.

**Specified By:**

`org.havi.ui.HNavigable.getGainFocusSound()` in interface `org.havi.ui.HNavigable`

**Returns:**

The sound played when the component gains focus. If no sound is associated with gaining focus, then null shall be returned.

---

**getLoseFocusSound()**

```
public org.havi.ui.HSound getLoseFocusSound()
```

Get the sound associated with the lost focus event.

**Specified By:**

`org.havi.ui.HNavigable.getLoseFocusSound()` in interface `org.havi.ui.HNavigable`

**Returns:**

The sound played when the component loses focus. If no sound is associated with losing focus, then null shall be returned.

---

**getMove(int)**

```
public org.havi.ui.HNavigable getMove(int keyCode)
```

Provides the `HNavigable` object that is navigated to when a particular key is pressed.

**Specified By:**

`org.havi.ui.HNavigable.getMove(int)` in interface `org.havi.ui.HNavigable`

**Parameters:**

`keyCode` - The key code of the pressed key.

**Returns:**

Returns the `HNavigable` object, or if no `HNavigable` is associated with the `keyCode` then returns null.

---

**getNavigationKeys()**

```
public int[] getNavigationKeys()
```

Retrieve the set of key codes which this component maps to navigation targets.

**Specified By:**

`org.havi.ui.HNavigationInputPreferred.getNavigationKeys()` in interface `org.havi.ui.HNavigationInputPreferred`

**Returns:**

an array of key codes, or null if no navigation targets are set on this component.

---

---

**isSelected()**

```
public boolean isSelected()
```

Indicates if this component has focus.

**Specified By:**

`org.havi.ui.HNavigable.isSelected()` in interface `org.havi.ui.HNavigable`

**Returns:**

`true` if the component has focus, otherwise returns `false`.

---

**processHFocusEvent(HFocusEvent)**

```
public void processHFocusEvent(org.havi.ui.event.HFocusEvent evt)
```

Process an `HFocusEvent` sent to this `HNavigationInputPreferred`.

**Specified By:**

`org.havi.ui.HNavigationInputPreferred.processHFocusEvent(org.havi.ui.event.HFocusEvent)` in interface `org.havi.ui.HNavigationInputPreferred`

**Parameters:**

`evt` - the `HFocusEvent` to process.

---

**removeInnerApplicationListener(InnerApplicationListener)**

```
public void removeInnerApplicationListener(InnerApplicationListener l)
```

Remove a listener for events when the inner application exits.

**Parameters:**

`l` - the listener to be removed

---

**setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)**

```
public void setFocusTraversal(org.havi.ui.HNavigable up, org.havi.ui.HNavigable down,  
    org.havi.ui.HNavigable left, org.havi.ui.HNavigable right)
```

Set the focus control for an `HNavigable` component. Note `setFocusTraversal` is a convenience function for application programmers where a standard up, down, left and right focus traversal between components is required.

Note `setFocusTraversal` is equivalent to multiple calls to `setMove`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

Note that this API does not prevent the creation of "isolated" `HNavigable` components --- authors should endeavor to avoid confusing the user.

**Specified By:**

`org.havi.ui.HNavigable.setFocusTraversal(org.havi.ui.HNavigable, org.havi.ui.HNavigable, org.havi.ui.HNavigable, org.havi.ui.HNavigable)` in interface `org.havi.ui.HNavigable`

**Parameters:**

`up` - The `HNavigable` component to move to, when the user generates a `VK_UP` `KeyEvent`. If there is no `HNavigable` component to move "up" to, then null should be specified.

`down` - The `HNavigable` component to move to, when the user generates a `VK_DOWN` `KeyEvent`. If there is no `HNavigable` component to move "down" to, then null should be specified.

`left` - The `HNavigable` component to move to, when the user generates a `VK_LEFT` `KeyEvent`. If there is no `HNavigable` component to move "left" to, then null should be specified.

---

`right` - The `HNavigable` component to move to, when the user generates a `VK_RIGHT` `KeyEvent`. If there is no `HNavigable` component to move "right" to, then null should be specified.

---

### **setGainFocusSound(HSound)**

```
public void setGainFocusSound(org.havi.ui.HSound sound)
```

Associate a sound with gaining focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_GAINED`. This sound will start to be played when an object implementing this interface gains focus. It is not guaranteed to be played to completion. If the object implementing this interface loses focus before the audio completes playing, the audio will be truncated. Applications wishing to ensure the audio is always played to completion must implement special logic to slow down the focus transitions.

By default, an `HNavigable` object does not have any gain focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

#### **Specified By:**

`org.havi.ui.HNavigable.setGainFocusSound(org.havi.ui.HSound)` in interface `org.havi.ui.HNavigable`

#### **Parameters:**

`sound` - the sound to be played, when the component gains focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

---

### **setLoseFocusSound(HSound)**

```
public void setLoseFocusSound(org.havi.ui.HSound sound)
```

Associate a sound with losing focus, i.e. when the `HNavigable` receives a `java.awt.event.FocusEvent` event of type `FOCUS_LOST`. This sound will start to be played when an object implementing this interface loses focus. It is not guaranteed to be played to completion. It is implementation dependent whether and when this sound will be truncated by any gain focus sound played by the next object to gain focus.

By default, an `HNavigable` object does not have any lose focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

#### **Specified By:**

`org.havi.ui.HNavigable.setLoseFocusSound(org.havi.ui.HSound)` in interface `org.havi.ui.HNavigable`

#### **Parameters:**

`sound` - the sound to be played, when the component loses focus. If sound content is already set, the original content is replaced. To remove the sound specify a null `HSound`.

---

### **setMove(int, HNavigable)**

```
public void setMove(int keyCode, org.havi.ui.HNavigable target)
```

Defines the navigation path from the current `HNavigable` to another `HNavigable` when a particular key is pressed.

Note that `setFocusTraversal` is equivalent to multiple calls to `setMove`, where the key codes `VK_UP`, `VK_DOWN`, `VK_LEFT`, `VK_RIGHT` are used.

#### **Specified By:**

`org.havi.ui.HNavigable.setMove(int, org.havi.ui.HNavigable)` in interface `org.havi.ui.HNavigable`



**Parameters:**

`keyCode` - The key code of the pressed key. Any numerical keycode is allowed, but the platform may not be able to generate all keycodes. Application authors should only use keys for which `HRcCapabilities.isSupported()` or `HKeyCapabilities.isSupported()` returns true.

`target` - The target `HNavigable` object that should be navigated to. If a target is to be removed from a particular navigation path, then `null` should be specified.

# org.dvb.application.inner InnerApplicationEvent

## Syntax

```
public class InnerApplicationEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.dvb.application.inner.InnerApplicationEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

InnerApplicationEvent is used to notify applications which have registered interest in events coming from inner applications that such an event has happened.

## Constructors

---

### InnerApplicationEvent(InnerApplicationContainer)

```
public InnerApplicationEvent(InnerApplicationContainer source)
```

Constructor for an event.

#### Parameters:

source - the InnerApplicationContainer which generated the event.

# org.dvb.application.inner InnerApplicationListener

## Syntax

```
public interface InnerApplicationListener
```

## Description

Listener for events related to inner applications

## Methods

---

### InnerApplicationExited(InnerApplicationEvent)

```
public void InnerApplicationExited(InnerApplicationEvent e)
```

Called when an inner application exits.

#### Parameters:

e - an event object encapsulating what happened

---

## **Annex AF (normative): Plug-in APIs**

# Package org.dvb.application.plugins

## Description

Provides support for inter-operable plugins in the MHP specification. The following classes and interfaces are intended for generic plug-ins and are not specific to any particular content format.

- InvalidApplicationException
- Plugin

The following classes are intended specifically for DVB-HTML plug-ins.

- ApplicationSecurityContext
- XletContainer
- XletSystemCall

Class Summary	
<b>Interfaces</b>	
<code>InvocationHandler</code>	<code>InvocationHandler</code> is the interface implemented by the <i>invocation handler</i> of a proxy instance.
<code>Plugin</code>	This interface allows an application or service manager to control the lifecycle of applications being executed by an inter-operable plug-in.
<b>Classes</b>	
<code>ApplicationSecurityContext</code>	This class represents the security context for an application managed by user code, e.g.
<code>Proxy</code>	<code>Proxy</code> provides static methods for creating dynamic proxy classes and instances, and it is also the superclass of all dynamic proxy classes created by those methods.
<code>XletContainer</code>	This class provides a container that can be embedded in a widget hierarchy, and can provide a container that can safely be used as the top-level container of an embedded Xlet.
<code>XletSystemCall</code>	This class permits user code to intercept certain system calls initiated by an embedded Xlet that need to be serviced by a support application.
<b>Exceptions</b>	
<code>InvalidApplicationException</code>	Thrown if an application is not valid for execution by a plug-in.
<code>UndeclaredThrowableException</code>	Thrown by a method invocation on a proxy instance if its invocation handler's <code>invoke(Object, Method, Object[])</code> method throws a checked exception (a <code>Throwable</code> that is not assignable to <code>RuntimeException</code> or <code>Error</code> ) that is not assignable to any of the exception types declared in the <code>throws</code> clause of the method that was invoked on the proxy instance and dispatched to the invocation handler.

# org.dvb.application.plugins ApplicationSecurityContext

## Syntax

```
public class ApplicationSecurityContext
    java.lang.Object
    |
    +--org.dvb.application.plugins.ApplicationSecurityContext
```

## Description

This class represents the security context for an application managed by user code, e.g. by a plug-in application managing DVB-HTML applications.

## Constructors

---

### ApplicationSecurityContext(URL[], Locator)

```
public ApplicationSecurityContext(java.net.URL[] path, javax.tv.locator.Locator entryPoint)
```

Creates a new security context for an application whose code can be found on the path supplied, and whose entry point is as given. Under no circumstances will an application managed by a plug-in be given access to resources not available to the plug-in itself. This policy is reflected in the permissions granted to the ApplicationSecurityContext, as well as the permissions granted to any classes loaded by any class loaders managed by a security context.

#### Parameters:

- `path` - The search path for locating resources within the application.
- `entryPoint` - The resource of the entry point of this application.

#### Throws:

- `NullPointerException` - if `entryPoint` is null, if `path` is null, or if any element of `path` is null.
- `IllegalArgumentException` - if `path.length < 1`

## Methods

---

### checkPermission(Permission)

```
public void checkPermission(java.security.Permission p)
```

Throws a `SecurityException` if the requested access, specified by the given permission, is not permitted to the application or sub-application represented by this application security context object. The set of permissions granted to an entity is a function of receiver policy, possibly influenced by user settings, application signer, and permission request file.

#### Parameters:

- `p` - A permission object representing the resource for which access is being checked.

#### Throws:

- `NullPointerException` - if `p` is null
- `SecurityException` - if this application has not been granted access to the resource represented by `p`.

---

**createEmbeddedContext(URL[], Locator)**

```
public ApplicationSecurityContext createEmbeddedContext(java.net.URL[] path,
    javax.tv.locator.Locator entryPoint)
```

Creates a context for an embedded part of an application, e.g. an Xlet embedded within a DVB-HTML page. The set of permissions granted to the application will be the same as the parent ApplicationSecurityContext.

**Parameters:**

- `path` - The search path for locating resources within the application.
- `entryPoint` - The resource of the entry point of this application.

**Returns:**

a context for the part of the application concerned

**Throws:**

- `NullPointerException` - if `entryPoint` is null, if `path` is null, or if any element of `path` is null.
- `IllegalArgumentException` - if `path.length < 1`

---

**getClassLoader(String[])**

```
public java.lang.ClassLoader getClassLoader(java.lang.String[] forbiddenPackages)
```

Get a classloader that is appropriate for loading classes for the application (or sub-application) represented by this application security context object. If this method is called more than once, the same instance will be returned.

It is important that embedded DVB-J code be prevented from accessing classes that implement the plug-in applicaiton. To this end, the plug-in may specify a list of forbidden packages. Classes loaded by the returned classloader will be forbidden from loading or accessing classes in the named packages.

**Parameters:**

- `forbiddenPackages` - a list of forbidden package names, e.g. { "de.tu-bs.ing.ifn.plugin.impl" }.

**Returns:**

A class loader that is appropriate for loading and DVB-J classes that are a part of this application or sub-application.

---

**getResource(String, boolean)**

```
public java.net.URL getResource(java.lang.String name, boolean sameSigner)
```

Get a locator to the named resource, within the search path for this application. Will return null if a resource with the given name that is appropriately signed (if necessary) cannot be found.

Note (informative): This method can be used, for example, by an interoperable plug-in that needs to fetch part of an application that is not loaded by a classloader. For example, it could be used to get a locator to an HTML page, if and only if that page is appropriately signed.

**Parameters:**

- `name` - The name of the resource (e.g. `com/foo/MyPage.html`)
- `sameSigner` - True if this is code, or any other resource for which the signer must be the same as the signer of the entry point.

**Returns:**

URL to the named resource, or null.

# org.dvb.application.plugins InvalidApplicationException

## Syntax

```
public class InvalidApplicationException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--org.dvb.application.plugins.InvalidApplicationException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Thrown if an application is not valid for execution by a plug-in.

## Since:

MHP1.1

## Constructors

---

### InvalidApplicationException()

```
public InvalidApplicationException()
```

Construct a InvalidApplicationException with no detail message

---

### InvalidApplicationException(String)

```
public InvalidApplicationException(java.lang.String s)
```

Construct a InvalidApplicationException with a detail message

#### Parameters:

*s* - detail message



# org.dvb.application.plugins InvocationHandler

## Syntax

```
public interface InvocationHandler
```

## Description

`InvocationHandler` is the interface implemented by the *invocation handler* of a proxy instance.

Each proxy instance has an associated invocation handler. When a method is invoked on a proxy instance, the method invocation is encoded and dispatched to the `invoke` method of its invocation handler.

## Methods

---

### `invoke(Object, Method, Object[])`

```
public java.lang.Object invoke(java.lang.Object proxy, java.lang.reflect.Method method,
                               java.lang.Object[] args)
```

Processes a method invocation on a proxy instance and returns the result. This method will be invoked on an invocation handler when a method is invoked on a proxy instance that it is associated with.

#### Parameters:

`proxy` - the proxy instance that the method was invoked on

`method` - the `Method` instance corresponding to the interface method invoked on the proxy instance. The declaring class of the `Method` object will be the interface that the method was declared in, which may be a superinterface of the proxy interface that the proxy class inherits the method through.

`args` - an array of objects containing the values of the arguments passed in the method invocation on the proxy instance, or `null` if interface method takes no arguments. Arguments of primitive types are wrapped in instances of the appropriate primitive wrapper class, such as `java.lang.Integer` or `java.lang.Boolean`.

#### Returns:

the value to return from the method invocation on the proxy instance. If the declared return type of the interface method is a primitive type, then the value returned by this method must be an instance of the corresponding primitive wrapper class; otherwise, it must be a type assignable to the declared return type. If the value returned by this method is `null` and the interface method's return type is primitive, then a `NullPointerException` will be thrown by the method invocation on the proxy instance. If the value returned by this method is otherwise not compatible with the interface method's declared return type as described above, a `ClassCastException` will be thrown by the method invocation on the proxy instance.

#### Throws:

`Throwable` - the exception to throw from the method invocation on the proxy instance. The exception's type must be assignable either to any of the exception types declared in the `throws` clause of the interface method or to the unchecked exception types `java.lang.RuntimeException` or `java.lang.Error`. If a checked exception is thrown by this method that is not assignable to any of the exception types declared in the `throws` clause of the interface method, then an `UndeclaredThrowableException` containing the exception that was thrown by this method will be thrown by the method invocation on the proxy instance.

#### See Also:

`UndeclaredThrowableException`

# org.dvb.application.plugins Plugin

## Syntax

```
public interface Plugin
```

## Description

This interface allows an application or service manager to control the lifecycle of applications being executed by an inter-operable plug-in. It shall be implemented by inter-operable plug-ins and shall be called by implementation of the application manager in an MHP terminal. It shall not be called by MHP applications. The usage model for this interface is the reuse of the `javax.tv.xlet.Xlet` interface to represent applications in formats other than DVB-J. It is the responsibility of the plug-in to provide implementations of the methods in that interface and to perform the translation between the semantics specified for DVB-J applications and the semantics of the application format which the plug-in supports. The plug-in is initialised by the application manager calling the no-argument constructor of the object implementing this interface. Plug-ins are not expected to perform any time consuming activities or hold any scarce resources in this call.

## Methods

---

### **initApplication(AppAttributes)**

```
public javax.tv.xlet.Xlet initApplication(org.dvb.application.AppAttributes app)
```

Request the plug-in to prepare for executing an instance of an application in a format which it supports. If the application cannot be prepared then null shall be returned. While the plug-in is in the terminated state then this method shall always return null. The application shall only be considered to be started when the application manager calls the methods on the Xlet interface returned by this method.

#### **Parameters:**

`app` - an instance of `AppAttributes` for the application which is to be started.

#### **Returns:**

an object implementing the Xlet interface or null if the application cannot be started.

#### **Throws:**

`InvalidApplicationException` - if the application to be started is not valid for this plug-in

---

### **initApplication(InnerApplication)**

```
public javax.tv.xlet.Xlet initApplication(InnerApplication app)
```

Request the plug-in to prepare for executing an instance of an application in a format which it supports. If the application cannot be prepared then null shall be returned. While the plug-in is in the terminated state then this method shall always return null. The application shall only be considered to be started when the application manager calls the methods on the Xlet interface returned by this method.

#### **Parameters:**

`app` - an instance of `InnerApplication` for the application which is to be started

#### **Throws:**

`InvalidApplicationException` - if the application to be started is not valid for this plug-in

---

### **initPlugin()**

```
public boolean initPlugin()
```

Initialise the plug-in. Any time consuming initialisations should be performed during this method call. This method shall be called after the no-argument constructor of the object implementing this interface and before any calls to the `startApplication`. It may also be called after a call to `terminatePlugin` if a plug-in was not removed from the virtual machine where it was executing. The behaviour of an application manager should a plug-in fail to initialise is intentionally unspecified.

**Returns:**

true if the plug-in initialised successfully otherwise false

---

### **isSupported(AppAttributes)**

```
public boolean isSupported(org.dvb.application.AppAttributes app)
```

Test whether this plug-in is able to support a particular application. This method shall work regardless of whether the plug-in has been initialised or not.

**Parameters:**

`app` - an instance of `AppAttributes` for the application which is to be started.

**Returns:**

true if the plug-in can support this application otherwise false

---

### **terminatePlugin()**

```
public void terminatePlugin()
```

Terminate the plug-in. This method shall only be called after all applications being executed by this plug-in have been terminated. The implementation of this method shall release all resources held by the plug-in. Once this method has returned, the plug-in shall either be unloaded from the virtual machine where it was executing or the `initPlugin` method shall be called again to put the plug-in again in an initialised condition. A plug-in may not refuse to terminate and throwing a runtime exception or error from this method shall result in the plug-in being removed from the virtual machine where it was executing.

# org.dvb.application.plugins Proxy

## Syntax

```
public class Proxy

java.lang.Object
|
+--org.dvb.application.plugins.Proxy
```

## Description

`Proxy` provides static methods for creating dynamic proxy classes and instances, and it is also the superclass of all dynamic proxy classes created by those methods.

To create a proxy for some interface `Foo`:

```
InvocationHandler handler = new MyInvocationHandler(...);
Class proxyClass = Proxy.getProxyClass(
    Foo.class.getClassLoader(), new Class[] { Foo.class });
Foo f = (Foo) proxyClass.
    getConstructor(new Class[] { InvocationHandler.class }).
    newInstance(new Object[] { handler });
```

or more simply:

```
Foo f = (Foo) Proxy.newProxyInstance(Foo.class.getClassLoader(),
    new Class[] { Foo.class },
    handler);
```

A *dynamic proxy class* (simply referred to as a *proxy class* below) is a class that implements a list of interfaces specified at runtime when the class is created, with behavior as described below. A *proxy interface* is such an interface that is implemented by a proxy class. A *proxy instance* is an instance of a proxy class. Each proxy instance has an associated *invocation handler* object, which implements the interface `InvocationHandler`. A method invocation on a proxy instance through one of its proxy interfaces will be dispatched to the `invoke(Object, Method, Object[])` method of the instance's invocation handler, passing the proxy instance, a `java.lang.reflect.Method` object identifying the method that was invoked, and an array of type `Object` containing the arguments. The invocation handler processes the encoded method invocation as appropriate and the result that it returns will be returned as the result of the method invocation on the proxy instance.

A proxy class has the following properties:

- Proxy classes are public, final, and not abstract.
- The unqualified name of a proxy class is unspecified. The space of class names that begin with the string "\$Proxy" should be, however, reserved for proxy classes.
- A proxy class extends `org.dvb.application.plugins.Proxy`.
- A proxy class implements exactly the interfaces specified at its creation, in the same order.
- If a proxy class implements a non-public interface, then it will be defined in the same package as that interface. Otherwise, the package of a proxy class is also unspecified. Note that package sealing will not prevent a proxy class from being successfully defined in a particular package at runtime, and neither will classes already defined in the same class loader and the same package with particular signers.
- Since a proxy class implements all of the interfaces specified at its creation, invoking `getInterfaces` on its `Class` object will return an array containing the same list of interfaces (in the order specified at its creation), invoking `getMethods` on its `Class` object will return an array of `Method` objects that include all of the methods in those interfaces, and invoking `getMethod` will find methods in the proxy interfaces as would be expected.
- The `isProxyClass(Class)` method will return true if it is passed a proxy class-- a class returned by `Proxy.getProxyClass` or the class of an object returned by `Proxy.newProxyInstance`-- and false otherwise.

- The `java.security.ProtectionDomain` of a proxy class is the same as that of system classes loaded by the bootstrap class loader, such as `java.lang.Object`, because the code for a proxy class is generated by trusted system code. This protection domain will typically be granted `java.security.AllPermission`.
- Each proxy class has one public constructor that takes one argument, an implementation of the interface `InvocationHandler`, to set the invocation handler for a proxy instance. Rather than having to use the reflection API to access the public constructor, a proxy instance can also be created by calling the `newProxyInstance(ClassLoader, Class[], InvocationHandler)` method, which combines the actions of calling `getProxyClass(ClassLoader, Class[])` with invoking the constructor with an invocation handler.

A proxy instance has the following properties:

- Given a proxy instance `proxy` and one of the interfaces implemented by its proxy class `Foo`, the following expression will return true:

```
proxy instanceof Foo
```

and the following cast operation will succeed (rather than throwing a `ClassCastException`):

```
(Foo) proxy
```

- Each proxy instance has an associated invocation handler, the one that was passed to its constructor. The static `getInvocationHandler(Object)` method will return the invocation handler associated with the proxy instance passed as its argument.
- An interface method invocation on a proxy instance will be encoded and dispatched to the invocation handler's `invoke(Object, Method, Object[])` method as described in the documentation for that method.
- An invocation of the `hashCode`, `equals`, or `toString` methods declared in `java.lang.Object` on a proxy instance will be encoded and dispatched to the invocation handler's `invoke` method in the same manner as interface method invocations are encoded and dispatched, as described above. The declaring class of the `Method` object passed to `invoke` will be `java.lang.Object`. Other public methods of a proxy instance inherited from `java.lang.Object` are not overridden by a proxy class, so invocations of those methods behave like they do for instances of `java.lang.Object`.
- **Methods Duplicated in Multiple Proxy Interfaces**

When two or more interfaces of a proxy class contain a method with the same name and parameter signature, the order of the proxy class's interfaces becomes significant. When such a *duplicate method* is invoked on a proxy instance, the `Method` object passed to the invocation handler will not necessarily be the one whose declaring class is assignable from the reference type of the interface that the proxy's method was invoked through. This limitation exists because the corresponding method implementation in the generated proxy class cannot determine which interface it was invoked through. Therefore, when a duplicate method is invoked on a proxy instance, the `Method` object for the method in the foremost interface that contains the method (either directly or inherited through a superinterface) in the proxy class's list of interfaces is passed to the invocation handler's `invoke` method, regardless of the reference type through which the method invocation occurred.

If a proxy interface contains a method with the same name and parameter signature as the `hashCode`, `equals`, or `toString` methods of `java.lang.Object`, when such a method is invoked on a proxy instance, the `Method` object passed to the invocation handler will have `java.lang.Object` as its declaring class. In other words, the public, non-final methods of `java.lang.Object` logically precede all of the proxy interfaces for the determination of which `Method` object to pass to the invocation handler.

Note also that when a duplicate method is dispatched to an invocation handler, the `invoke` method may only throw checked exception types that are assignable to one of the exception types in the `throws` clause of the method in *all* of the proxy interfaces that it can be invoked through. If the `invoke` method throws a checked exception that is not assignable to any of the exception types declared by the method in one of the proxy interfaces that it can be invoked through, then an unchecked `UndeclaredThrowableException` will be thrown by the invocation on the proxy instance. This restriction means that not all of the exception types returned by invoking `getExceptionTypes` on the `Method` object passed to the `invoke` method can necessarily be thrown successfully by the `invoke` method.

## Fields

---

### h

protected `InvocationHandler` h

the invocation handler for this proxy instance.

## Constructors

---

### Proxy(InvocationHandler)

protected `Proxy(InvocationHandler h)`

Constructs a new `Proxy` instance from a subclass (typically, a dynamic proxy class) with the specified value for its invocation handler.

**Parameters:**

h - the invocation handler for this proxy instance

## Methods

---

### getInvocationHandler(Object)

public static `InvocationHandler` getInvocationHandler(`java.lang.Object` proxy)

Returns the invocation handler for the specified proxy instance.

**Parameters:**

proxy - the proxy instance to return the invocation handler for

**Returns:**

the invocation handler for the proxy instance

**Throws:**

`IllegalArgumentException` - if the argument is not a proxy instance

---

### getProxyClass(ClassLoader, Class[])

public static `java.lang.Class` getProxyClass(`java.lang.ClassLoader` loader,  
`java.lang.Class[]` interfaces)

Returns the `java.lang.Class` object for a proxy class given a class loader and an array of interfaces. The proxy class will be defined in the specified class loader and will implement all of the supplied interfaces. If a proxy class for the same permutation of interfaces has already been defined in the class loader, then the existing proxy class will be returned; otherwise, a proxy class for those interfaces will be generated dynamically and defined in the class loader.

There are several restrictions on the parameters that may be passed to `Proxy.getProxyClass`:

- All of the `Class` objects in the `interfaces` array must represent interfaces, not classes or primitive types.
- No two elements in the `interfaces` array may refer to identical `Class` objects.
- All of the interface types must be visible by name through the specified class loader. In other words, for class loader `cl` and every interface `i`, the following expression must be true:

```
Class.forName(i.getName(), false, cl) == i
```

- All non-public interfaces must be in the same package; otherwise, it would not be possible for the proxy class to implement all of the interfaces, regardless of what package it is defined in.
- No two interfaces may each have a method with the same name and parameter signature but different return type.
- The resulting proxy class must not exceed any limits imposed on classes by the virtual machine. For example, the VM may limit the number of interfaces that a class may implement to 65535; in that case, the size of the `interfaces` array must not exceed 65535.

If any of these restrictions are violated, `Proxy.getProxyClass` will throw an `IllegalArgumentException`. If the `interfaces` array argument or any of its elements are null, a `NullPointerException` will be thrown.

Note that the order of the specified proxy interfaces is significant: two requests for a proxy class with the same combination of interfaces but in a different order will result in two distinct proxy classes.

**Parameters:**

`loader` - the class loader to define the proxy class in

`interfaces` - the list of interfaces for the proxy class to implement

**Returns:**

a proxy class that is defined in the specified class loader and that implements the specified interfaces

**Throws:**

`IllegalArgumentException` - if any of the restrictions on the parameters that may be passed to `getProxyClass` are violated

`NullPointerException` - if the `interfaces` array argument or any of its elements are null

### **isProxyClass(Class)**

```
public static boolean isProxyClass(java.lang.Class cl)
```

Returns true if and only if the specified class was dynamically generated to be a proxy class using the `getProxyClass` method or the `newProxyInstance` method.

The reliability of this method is important for the ability to use it to make security decisions, so its implementation should not just test if the class in question extends `Proxy`.

**Parameters:**

`cl` - the class to test

**Returns:**

`true` if the class is a proxy class and `false` otherwise

**Throws:**

`NullPointerException` - if `cl` is null

### **newProxyInstance(ClassLoader, Class[], InvocationHandler)**

```
public static java.lang.Object newProxyInstance(java.lang.ClassLoader loader,
        java.lang.Class[] interfaces, InvocationHandler h)
```

Returns an instance of a proxy class for the specified interfaces that dispatches method invocations to the specified invocation handler. This method is equivalent to:

```
Proxy.getProxyClass(loader, interfaces).
    getConstructor(new Class[] { InvocationHandler.class }).
    newInstance(new Object[] { handler });
```

`Proxy.newProxyInstance` throws `IllegalArgumentException` for the same reasons that `Proxy.getProxyClass` does.

**Parameters:**

`loader` - the class loader to define the proxy class in

`interfaces` - the list of interfaces for the proxy class to implement

`h` - the invocation handler to dispatch method invocations to

**Returns:**

a proxy instance with the specified invocation handler of a proxy class that is defined in the specified class loader and that implements the specified interfaces

**Throws:**

`IllegalArgumentException` - if any of the restrictions on the parameters that may be passed to `getProxyClass` are violated

`NullPointerException` - if the `interfaces` array argument or any of its elements are null, or if the invocation handler, `h`, is null



# org.dvb.application.plugins UndeclaredThrowableException

## Syntax

```
public class UndeclaredThrowableException extends java.lang.RuntimeException
```

```
java.lang.Object
|
+--java.lang.Throwable
|
+--java.lang.Exception
|
+--java.lang.RuntimeException
|
+--org.dvb.application.plugins.UndeclaredThrowableException
```

## All Implemented Interfaces:

```
java.io.Serializable
```

## Description

Thrown by a method invocation on a proxy instance if its invocation handler's `invoke(Object, Method, Object[])` method throws a checked exception (a `Throwable` that is not assignable to `RuntimeException` or `Error`) that is not assignable to any of the exception types declared in the `throws` clause of the method that was invoked on the proxy instance and dispatched to the invocation handler.

An `UndeclaredThrowableException` instance contains the undeclared checked exception that was thrown by the invocation handler, and it can be retrieved with the `getUndeclaredThrowable()` method. `UndeclaredThrowableException` extends `RuntimeException`, so it is an unchecked exception that wraps a checked exception.

## Constructors

---

### UndeclaredThrowableException(Throwable)

```
public UndeclaredThrowableException(java.lang.Throwable undeclaredThrowable)
```

Constructs an `UndeclaredThrowableException` with the specified `Throwable`.

#### Parameters:

`undeclaredThrowable` - the undeclared checked exception that was thrown

---

### UndeclaredThrowableException(Throwable, String)

```
public UndeclaredThrowableException(java.lang.Throwable undeclaredThrowable,
    java.lang.String s)
```

Constructs an `UndeclaredThrowableException` with the specified `Throwable` and a detail message.

#### Parameters:

`undeclaredThrowable` - the undeclared checked exception that was thrown

`s` - the detail message

## Methods

---

### **getUndeclaredThrowable()**

```
public java.lang.Throwable getUndeclaredThrowable()
```

Returns the `Throwable` instance wrapped in this `UndeclaredThrowableException`.

**Returns:**

the undeclared checked exception that was thrown

---

### **printStackTrace()**

```
public void printStackTrace()
```

Prints this `UndeclaredThrowableException` and its backtrace to the standard error stream.

**Overrides:**

`java.lang.Throwable.printStackTrace()` in class `java.lang.Throwable`

---

### **printStackTrace(PrintStream)**

```
public void printStackTrace(java.io.PrintStream ps)
```

Prints this `UndeclaredThrowableException` and its backtrace to the specified `PrintStream`.

**Overrides:**

`java.lang.Throwable.printStackTrace(java.io.PrintStream)` in class `java.lang.Throwable`

**Parameters:**

`ps` - the `PrintStream` to print the exception to

---

### **printStackTrace(PrintWriter)**

```
public void printStackTrace(java.io.PrintWriter pw)
```

Prints this `UndeclaredThrowableException` and its backtrace to the specified `PrintWriter`.

**Overrides:**

`java.lang.Throwable.printStackTrace(java.io.PrintWriter)` in class `java.lang.Throwable`

**Parameters:**

`pw` - the `PrintWriter` to print this exception to

# org.dvb.application.plugins XletContainer

## Syntax

```
public class XletContainer extends java.awt.Container
```

```
java.lang.Object
|
+--java.awt.Component
   |
   +--java.awt.Container
      |
      +--org.dvb.application.plugins.XletContainer
```

## All Implemented Interfaces:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

## Description

This class provides a container that can be embedded in a widget hierarchy, and can provide a container that can safely be used as the top-level container of an embedded Xlet. This solves the problem of an embedded Xlet being able to call `getParent()` to discover the widget hierarchy in which it is embedded.

## Since:

MHP 1.1

## Constructors

---

### XletContainer(Container)

```
public XletContainer(java.awt.Container parent)
```

Construct a new XletContainer as a child of Parent. The XletContainer contains exactly one widget: A child container. The child container is constrained to be at the same location and the same size as the parent XletContainer.

#### Parameters:

parent - The widget that is to have an Xlet embedded within it.

#### Throws:

NullPointerException - if parent is null

#### See Also:

[getXletContainer\(\)](#)

## Methods

---

### getXletContainer()

```
public java.awt.Container getXletContainer()
```

Get the only child of the XletContainer. This container can safely be used as the top-level container of an Xlet, e.g. an Xlet embedded in a DVB-HTML page. Calling getParent() on this container from client code will return null, unless the caller has been granted special access via a platform-dependent mechanism.

If this method is invoked more than once for the same instance of XletContainer, it will return the same instance.

**Returns:**

The child of this XletContainer.

# org.dvb.application.plugins XletSystemCall

## Syntax

```
public abstract class XletSystemCall  
  
java.lang.Object  
|  
+--org.dvb.application.plugins.XletSystemCall
```

## Description

This class permits user code to intercept certain system calls initiated by an embedded Xlet that need to be serviced by a support application. For example, a DVB-HTML plug-in application needs to service requests that are made by an embedded Xlet, typically via static method calls.

## Since:

MHP1.1

## Constructors

---

### XletSystemCall()

```
protected XletSystemCall()  
Create a new XletSystemCall
```

## Methods

---

### getRootContainer(XletContext)

```
public abstract java.awt.Container getRootContainer(javax.tv.xlet.XletContext ctx)
```

Called when the Xlet calls `javax.tv.graphics.TVContainer.getRootContainer()`.

#### Parameters:

`ctx` - The context of the Xlet making the request; it shall be identical to the `XletContext` used to create this instance of `XletSystemCall`.

#### Returns:

a container object to be returned to the embedded xlet

#### See Also:

`javax.tv.graphics.TVContainer`

---

### register(Plugin, XletContext)

```
public final void register(Plugin p, javax.tv.xlet.XletContext ctx)
```

Register this instance of `XletSystemCall` with the system.

**Parameters:**

`p` - The Plugin that services calls made by the xlet, i.e. the Plugin of which this instance of `XletSystemCall` is a part.

`ctx` - The `XletContext` of the Xlet making the calls

**Throws:**

`NullPointerException` - if `p` or `ctx` is null

**See Also:**

`unregister(Plugin, XletContext)`

---

**unregister(Plugin, XletContext)**

```
public final void unregister(Plugin p, javax.tv.xlet.XletContext ctx)
```

Unregister this instance of `XletSystemCall` with the system. When an interoperable Plugin terminates, or an Xlet managed by a Plugin is asked to terminate, the Plugin must unregister any relevant `XletSystemCall` instances.

**Parameters:**

`p` - The Plugin that services calls made by the xlet, i.e. the Plugin of which this instance of `XletSystemCall` is a part.

`ctx` - The `XletContext` of the Xlet making the calls

**Throws:**

`NullPointerException` - if `p` or `ctx` is null

**See Also:**

`register(Plugin, XletContext)`

---

## **Annex AG (normative): Stored application APIs**

The current draft of this package is a partial definition of the org.dvb.application package. It includes changed classes or interfaces, new classes or interfaces and some existing classes & interfaces which have not changed. These existing classes / interfaces which are included but which have not changed can be identified by the absence of any comment or description apart from the class or interface signature. It is an open editing issue to discover whether they can easily be removed or not.

# Package org.dvb.application.storage

## Description

Provides support for storage of applications.

Class Summary	
<b>Interfaces</b>	
<code>ApplicationStorage</code>	An interface that shall be implemented by the objects that implement <code>javax.service.Service</code> interface and where the MHP terminal supports storing applications.
<code>ExtendedAppAttributes</code>	The <code>ExtendedAppAttributes</code> interface provides additional attributes that are useful when application can be stored in the MHP terminal.
<code>StoredApplicationService</code>	Defines the information about a stored application service.
<code>StoredServiceCreator</code>	This interface supports creating new <code>Service</code> objects representing stand-alone stored application services.
<b>Classes</b>	
<code>ApplicationStoragePermission</code>	This class represents a permission to manage applications stored in the MHP terminal.
<b>Exceptions</b>	
<code>ApplicationDownloadException</code>	Thrown if the downloading of the application failed.
<code>InvalidApplicationException</code>	Thrown if an application is not valid for installing into a particular service.
<code>InvalidDescriptionFileException</code>	Thrown if the description of the application to be stored is invalid or otherwise not conformant to the specification.
<code>NotEnoughResourcesException</code>	Thrown if the MHP terminal does not have enough resources to install the application.
<code>ServiceAlreadyExistsException</code>	Thrown if a <code>StoredApplicationService</code> already exists with the identifiers the application tried to create a new one.
<code>UserRejectedInstallException</code>	Thrown if the end user rejects the request to install an application.



# org.dvb.application.storage ApplicationDownloadException

## Syntax

```
public class ApplicationDownloadException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--org.dvb.application.storage.ApplicationDownloadException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Thrown if the downloading of the application failed.

## Since:

MHP1.1

## Constructors

---

### ApplicationDownloadException()

```
public ApplicationDownloadException()
```

Constructs an ApplicationDownloadException with no detail message.

---

### ApplicationDownloadException(String)

```
public ApplicationDownloadException(java.lang.String s)
```

Constructs an ApplicationDownloadException with a detail message

#### Parameters:

*s* - detail message

# org.dvb.application.storage

# ApplicationStorage

## Syntax

```
public interface ApplicationStorage extends javax.tv.service.Service
```

## All Known Subinterfaces:

```
    StoredApplicationService
```

## All Superinterfaces:

```
    javax.tv.service.Service
```

## Description

An interface that shall be implemented by the objects that implement `javax.service.Service` interface and where the MHP terminal supports storing applications. MHP terminals which do not support storing applications shall not return objects implementing this interface.

When this object represents a service that is retrieved from a network, e.g. a broadcast service, storing applications using the methods in this interface will install them in the receiver as broadcast service related stored applications.

When this object represents a stored application service, storing applications using the methods in this interface will add the application into this service.

## Since:

MHP1.1

## Methods

---

### getStoredAppIDs()

```
public org.dvb.application.AppID[] getStoredAppIDs()
```

Lists the AppIDs of the applications that are stored within this service.

#### Returns:

an array of AppID object representing the stored application

#### Throws:

`SecurityException` - Thrown for broadcast services if the service represented by this object is not the currently selected service of the the service context in which the calling application is running. Thrown for a stand-alone stored service if the application calling this method does not have the same `organisation_id` as the application which originally created the stand-alone stored service.

#### Since:

MHP1.1

---

### getVersionNumber(AppID)

```
public int getVersionNumber(org.dvb.application.AppID appId)
```

Return the version number of the stored application whose AppID is given as a parameter.

**Parameters:**

`appId` - the AppID of the application whose version is queried

**Returns:**

the version number of the stored application, returns -1 if the application given as a parameter is not stored

**Throws:**

`SecurityException` - Thrown for broadcast services if the service represented by this object is not the currently selected service of the the service context in which the calling application is running. Thrown for a stand-alone stored service if the application calling this method does not have the same `organisation_id` as the application which originally created the stand-alone stored service.

**Since:**

MHP1.1

**initiateApplicationInstall(AppProxy, boolean, String[])**

```
public void initiateApplicationInstall(org.dvb.application.AppProxy app, boolean autoStart,
    java.lang.String[] args)
```

Requests the MHP terminal to initiate the installation of an application into the MHP terminal.

Applications should be prepared for the platform consulting the end user of the MHP terminal for the permission to install the application

Note: This method is synchronous and will block until the installation is either completed or fails.

**Parameters:**

`app` - an AppProxy representing the application to be installed

`autoStart` - true, if the application becomes an autostart application in the stored application service; false, if the application becomes a normal present (non-autostart) application in the stored application service. This parameter is not used if storing a broadcast service related application.

`args` - parameters to be available to the application when started. Passing in either null or an array of size zero indicates no parameters are to be available. These parameters shall be available to applications when running as part of the stored application service using the Xlet property "dvb.installer.parameters".

**Throws:**

`InvalidApplicationException` - thrown if the application is not valid for installation into this service. For a broadcast service, only applications signalled in that service are valid. For a stand-alone stored service, all applications identified as able to run stand-alone in their application description file are defined to be valid for the purposes of this exception.

`UserRejectedInstallException` - thrown if the end user rejects the installation

`NotEnoughResourcesException` - thrown if the MHP terminal does not have enough resources, e.g. storage space, available for the application

`InvalidDescriptionFileException` - thrown if the application description file is not valid

`ApplicationDownloadException` - thrown if the downloading of the application files was not successful

`java.lang.SecurityException` - Thrown if the application calling this method does not have an `ApplicationStoragePermission` with action "store" or "\*" for the `organisation_id` of the application to be stored. Also thrown for a stand-alone stored service if the application calling this method does not have the same `organisation_id` as the application which originally created the stand-alone stored service.

**Since:**

MHP1.1

---

**initiateApplicationRemoval(AppID)**

```
public void initiateApplicationRemoval(org.dvb.application.AppID app)
```

Requests the MHP terminal to initiate the removal of an application stored in the MHP terminal from this service.

Applications should be prepared for the platform consulting the end user of the MHP terminal for the permission to remove the application

If the application identified by the AppID passed in as a parameter is not installed in this service, the method shall fail silently.

**Parameters:**

app - AppID of the application to be removed

**Throws:**

`java.lang.SecurityException` - thrown if the application calling this method does not have an `ApplicationStoragePermission` with action "remove" or "\*" for the `organisation_id` of the application to be removed. Also thrown for a stand-alone stored service if the application calling this method does not have the same `organisation_id` as the application which originally created the stand-alone stored service.

**Since:**

MHP1.1

**See Also:**[ApplicationStoragePermission](#)

# org.dvb.application.storage ApplicationStoragePermission

## Syntax

```
public class ApplicationStoragePermission extends java.security.BasicPermission
```

```
java.lang.Object
|
+--java.security.Permission
|
+--java.security.BasicPermission
|
+--org.dvb.application.storage.ApplicationStoragePermission
```

## All Implemented Interfaces:

java.security.Guard, java.io.Serializable

## Description

This class represents a permission to manage applications stored in the MHP terminal. An ApplicationStoragePermission contains a name representing the organisation\_id whose applications can be managed and an actions list representing the permitted actions, i.e. store and/or remove applications.

The name of the permission contains the organisation\_id represented in hexadecimal form without any "0x" prefix. When the application possesses this permission, it is able to initiate storage or removal of applications signalled with this organisation\_id. If the application does not have the ApplicationStoragePermission granted with the same organisation\_id as the application is signalled with when it tries to initiate either storing or removal of an application, the methods to store/remove will throw a SecurityException.

The actions string shall be one of the following:-

- "store", representing permission to store an application
- "remove", representing permission to remove an application
- "create", representing permission to create a stored application service
- "remove\_service", representing permission to remove a stored application service
- "\*", representing all of the above

## Since:

MHP1.1

## Constructors

---

### ApplicationStoragePermission(String)

```
public ApplicationStoragePermission(java.lang.String name)
```

Creates a new ApplicationStoragePermission with the specified name. Permissions created with this constructor implicitly have the actions string set to "\*".

#### Parameters:

name - the organisation\_id whose applications can be managed. This is encoded in hexadecimal representation without any preceding "0x" prefix.

---

**ApplicationStoragePermission(String, String)**

```
public ApplicationStoragePermission(java.lang.String name, java.lang.String actions)
```

Creates a new ApplicationStoragePermission object with the specified name. The name contains the organisation\_id of the applications that can be managed and the actions String shall be either "store", "remove" or "\*". Permission objects constructed with incorrectly encoded parameters do not represent any permission and are ignored by the platform.

**Parameters:**

`name` - the organisation\_id whose applications can be managed. This is encoded in hexadecimal representation without any preceding "0x" prefix.

`actions` - Shall contain "store", "remove" or "\*"

# org.dvb.application.storage ExtendedAppAttributes

## Syntax

```
public interface ExtendedAppAttributes
```

## Description

The `ExtendedAppAttributes` interface provides additional attributes that are useful when application can be stored in the MHP terminal.

`AppAttributes` objects that are returned from the `AppsDatabase` shall implement this interface

## Since:

MHP1.1

## Methods

---

### `getCurrentVersionNumber()`

```
public int getCurrentVersionNumber()
```

Returns the optional version number currently signalled for this application. e.g. in the AIT. If this is not available, -1 shall be returned.

#### Returns:

the version number signalled for this application.

#### Since:

MHP1.1

# org.dvb.application.storage InvalidApplicationException

## Syntax

```
public class InvalidApplicationException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--org.dvb.application.storage.InvalidApplicationException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Thrown if an application is not valid for installing into a particular service.

## Since:

MHP1.1

## Constructors

---

### InvalidApplicationException()

```
public InvalidApplicationException()
```

Constructs an InvalidApplicationException with no detail message.

---

### InvalidApplicationException(String)

```
public InvalidApplicationException(java.lang.String s)
```

Construct a InvalidApplicationException with a detail message

#### Parameters:

*s* - detail message



# org.dvb.application.storage InvalidDescriptionFileException

## Syntax

```
public class InvalidDescriptionFileException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--org.dvb.application.storage.InvalidDescriptionFileException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Thrown if the description of the application to be stored is invalid or otherwise not conformant to the specification.

## Since:

MHP1.1

## Constructors

---

### InvalidDescriptionFileException()

```
public InvalidDescriptionFileException()
```

Constructs an InvalidDescriptionFileException with no detail message.

---

### InvalidDescriptionFileException(String)

```
public InvalidDescriptionFileException(java.lang.String s)
```

Constructs an InvalidDescriptionFileException with a detail message

#### Parameters:

s - detail message

# org.dvb.application.storage NotEnoughResourcesException

## Syntax

```
public class NotEnoughResourcesException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
|
+--java.lang.Exception
|
+--org.dvb.application.storage.NotEnoughResourcesException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Thrown if the MHP terminal does not have enough resources to install the application. e.g. not enough storage space.

## Since:

MHP1.1

## Constructors

---

### NotEnoughResourcesException()

```
public NotEnoughResourcesException()
```

Constructs a NotEnoughResourcesException with no detail message.

---

### NotEnoughResourcesException(String)

```
public NotEnoughResourcesException(java.lang.String s)
```

Constructs a NotEnoughResourcesException with a detail message

#### Parameters:

s - detail message

# org.dvb.application.storage ServiceAlreadyExistsException

## Syntax

```
public class ServiceAlreadyExistsException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--org.dvb.application.storage.ServiceAlreadyExistsException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Thrown if a StoredApplicationService already exists with the identifiers the application tried to create a new one.

## Since:

MHP1.1

## Constructors

---

### ServiceAlreadyExistsException()

```
public ServiceAlreadyExistsException()
```

Constructs a ServiceAlreadyExistsException with no detail message.

---

### ServiceAlreadyExistsException(String)

```
public ServiceAlreadyExistsException(java.lang.String s)
```

Constructs a ServiceAlreadyExistsException with a detail message

#### Parameters:

s - detail message

# org.dvb.application.storage StoredApplicationService

## Syntax

```
public interface StoredApplicationService extends javax.tv.service.Service, ApplicationStorage
```

## All Superinterfaces:

[ApplicationStorage](#), [javax.tv.service.Service](#)

## Description

Defines the information about a stored application service.

Stored application services can be created by applications that have a permission to store applications for an `organisation_id`.

Stored application services are uniquely identified within the terminal by the combination of the `organisation_id` and `service_id`.

## Since:

MHP1.1

## Fields

---

### STORED

```
public static final javax.tv.service.ServiceInformationType STORED
```

The service information is from storage

---

### STORED\_APPLICATION\_SERVICE

```
public static final javax.tv.service.ServiceType STORED_APPLICATION_SERVICE
```

The service type for a stored application

#### See Also:

[javax.tv.service.navigation.ServiceTypeFilter](#)

## Methods

---

### getLocator()

```
public javax.tv.locator.Locator getLocator()
```

Gets the Locator for this stored application service. This locator is opaque and platform specific. It is not required to be an instance of any public locator class in the platform.

#### Overrides:

[javax.tv.service.Service.getLocator\(\)](#) in interface [javax.tv.service.Service](#)

**Returns:**

a locator for this stored application service

---

**getName()**

```
public java.lang.String getName()
```

Returns the name of the stored application service as defined when the stored application service was created.

**Overrides:**

javax.tv.service.Service.getName() in interface javax.tv.service.Service

**Returns:**

the name of the stored application service

---

**getOrganisationId()**

```
public int getOrganisationId()
```

Return the organisation id of this stored application service

**Returns:**

the organisation id of this stored application service

---

**getServiceId()**

```
public int getServiceId()
```

Return the service id of this stored application service

**Returns:**

the service id of this stored application service

---

**getServiceInformationType()**

```
public javax.tv.service.ServiceInformationType getServiceInformationType()
```

Returns the service information format of this object. This shall always return STORED.

**Returns:**

the service information format

---

**getServiceType()**

```
public javax.tv.service.ServiceType getServiceType()
```

Returns the type of this service. For stored application services, this method shall always return STORED\_APPLICATION\_SERVICE.

**Overrides:**

javax.tv.service.Service.getServiceType() in interface javax.tv.service.Service

**Returns:**

service type of this service

---

**hasMultipleInstances()**

```
public boolean hasMultipleInstances()
```

Indicates whether the service represented by this Service is available on multiple transports. For stored application services, this shall always return false.

**Overrides:**

`javax.tv.service.Service.hasMultipleInstances()` in interface `javax.tv.service.Service`

**Returns:**

false

---

**removeService()**

```
public void removeService()
```

Request the removal of the whole stored application service from the terminal. Removal of the service results in the removal of all the applications from within that service.

---

**retrieveDetails(SIRequestor)**

```
public javax.tv.service.SIRequest retrieveDetails(javax.tv.service.SIRequestor requestor)
```

This method retrieves additional information about the service. This shall result in failure with the `SIRequestFailureType DATA_UNAVAILABLE`.

**Overrides:**

`javax.tv.service.Service.retrieveDetails(javax.tv.service.SIRequestor)` in interface `javax.tv.service.Service`

**Parameters:**

`requestor` - The `SIRequestor` to be notified when this operation completes.

**Returns:**

A `SIRequest` object identifying this request.

# org.dvb.application.storage

# StoredServiceCreator

## Syntax

```
public interface StoredServiceCreator
```

## Description

This interface supports creating new Service objects representing stand-alone stored application services. Services thus created shall appear in the list of services maintained by the SIManager until removed using `StoredApplicationService.remove` or some MHP terminal specific mechanism. i.e. they shall be returned by `filterServices` both when passed an instance of `ServiceTypeFilter` constructed with the type `StoredApplicationService.STORED_APPLICATION_SERVICE` and when passed null to list all known services.

This interface shall be implemented by the object returned from `javax.tv.service.SIManager.getInstance` only in those MHP implementations which support stand-alone stored applications.

## Since:

MHP1.1

## Methods

---

### **createStoredApplicationService(int, int, String)**

```
public StoredApplicationService createStoredApplicationService(int organisation_id,  
    int service_id, java.lang.String serviceName)
```

Requests the platform to create a new stored application service.

#### Parameters:

`organisation_id` - the `organisation_id` of the organisation to whom this service belongs to

`service_id` - unique identifier for this service within the organisation

`serviceName` - a name for the service that can be displayed to the end user to identify this service

#### Returns:

the stored application service created

#### Throws:

`UserRejectedInstallException` - thrown if the end user rejects the installation

`NotEnoughResourcesException` - thrown if the MHP terminal does not have enough resources, e.g. storage space, available for the application

`ServiceAlreadyExistsException` - thrown if a stored application service with the same `organisation_id` and `service_id` already exists in the terminal

# org.dvb.application.storage UserRejectedInstallException

## Syntax

```
public class UserRejectedInstallException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--org.dvb.application.storage.UserRejectedInstallException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Thrown if the end user rejects the request to install an application.

## Since:

MHP1.1

## Constructors

---

### UserRejectedInstallException()

```
public UserRejectedInstallException()
```

Constructs a UserRejectedInstallException with no detail message.

---

### UserRejectedInstallException(String)

```
public UserRejectedInstallException(java.lang.String s)
```

Constructs a UserRejectedInstallException with a detail message

#### Parameters:

*s* - detail message



---

## **Annex AH (normative): Internet client APIs**

# Package org.dvb.internet

## Description

Provides a mechanism for MHP applications to control internet clients that may be present on an MHP such as a web browser or email client.

This package provides a mechanism for MHP applications to control internet clients that may be present on an MHP such as a web browser or email client. The API makes no assumptions about whether an internet client and downloaded MHP application can run simultaneously, and in some cases starting an internet client may result in the calling application being killed.

The API consists of two main class hierarchies. `InternetClientService` and its clients provide a mechanism to get a locator for a resident internet client so that it can be started, and also provide any functionality that does not require the client to be started. This can include adding bookmarks or getting the capabilities of the clients. The second major part of the API is the `InternetClient` class and its subclasses. These provide an interface to a running instance of an internet client and support the operations that can only be carried out when the client is running.

The internet client API uses the JavaTV service navigation API for accessing resident applications. Any effect that the invocation of a resident internet client has on the lifecycle of a downloaded application is dependent on the service context in which the resident application is started (using `ServiceContext.select()`). If the internet client is started in the same service context as the downloaded application, then the downloaded application will be terminated, following the usual semantics of the service selection API. The following section introduces the basic concepts and then explores a few scenarios to illustrate combinations or broadcast application versus resident application execution.

## Concepts

The first implication of resident applications is that it should be feasible for broadcast applications to isolate, browse, and select such services. The collection of `javax.tv.service.*` packages provides the framework for the solution to this requirement.

The class `ServiceType` in the `javax.tv.service` package defines various kinds of services. This is extended in this API to provide additional service types for Internet clients. The `javax.tv` framework lets a client filter the services known to a platform to create a `ServiceCollection`. The client would first create an instance of `ServiceTypeFilter`

```
public ServiceTypeFilter(ServiceType type)
```

where the `ServiceType` in the signature would be the instance for the resident service in question. The client would then filter the services known to the platform to isolate the resident services of this type. The `ServiceCollection` interface provides the filter operation.

```
public ServiceCollection createServiceCollection(ServiceFilter filter)
```

Given the `ServiceCollection`, the client can iterate over the resident services to find a specific resident service. Given the `Service`, the client can recover its locator:

```
public Locator getLocator()
```

The last step is to `select()` the resident service. The `ServiceContext` interface of the `javax.tv.service.selection` package provides the operation:

```
public void select(Locator selection)
throws InvalidLocatorException, java.lang.SecurityException,
java.lang.IllegalStateException
```

The semantics of `select()` are that the implementation first examines the locator of the signature. If the locator is found to be valid, the operation returns. The implementation does not stall while it attempts to allocate resources to download the service, and then attempts to allocate resources to execute the service.

If the implementation determines that it can not realize the service due to scarce resources, it forwards a `SelectionFailedEvent` (of the `javax.tv.service.selection` package) with the reason code `INSUFFICIENT_RESOURCES`. The `ServiceContextListener` interface fields the event:

```
public void receiveServiceContextEvent(ServiceContextEvent e)
```

Thus the client should implement an instance of the `ServiceContextListener` interface and register interest in service context events. The `ServiceContext` interface provides the mechanism:

```
public void addListener(ServiceContextListener listener)
throws java.lang.IllegalStateException
```

The semantics of a `ServiceContext` are that at most a single `Service` can execute within a `ServiceContext` at a time. Since the premise is that a broadcast service selects a resident service, there are two scenarios of interest.

The first scenario presumes that the broadcast service wants to select the resident service but not survive. It expects the selection to cause its termination. The broadcast service in this case should create an instance of `ServiceContextFactory`. The operation on `ServiceContextFactory` itself is:

```
public static ServiceContextFactory getInstance()
```

The broadcast application then requests the service contexts that are visible to the broadcast application:

```
public abstract ServiceContext getServiceContext(XletContext ctx)
```

which returns *its* `ServiceContext`. The broadcast application selects the resident application within its service context:

```
public void select(Locator selection)
throws InvalidLocatorException, java.lang.SecurityException,
java.lang.IllegalStateException
```

which causes the broadcast application to terminate.

The premise of the second scenario is that the broadcast application wants to survive the selection of the resident service. In this case the broadcast again creates a `ServiceContextFactory` but then creates a `ServiceContext` which is not its `ServiceContext`:

```
public abstract ServiceContext createServiceContext()
throws InsufficientResourcesException,
java.lang.SecurityException
```

The broadcast application selects the resident application. Since the operation is on a `ServiceContext` that is not its `ServiceContext`, the broadcast application survives the selection.

One subtle aspect of the invocation patterns is that the platform does not know whether the broadcast application will attempt to select a resident application versus another broadcast application until the broadcast application invokes the `select()` operation. For example, assume the platform supports a single broadcast application plus a single resident application. Further assume that, after the broadcast application creates a second `ServiceContext`, it selects a second broadcast application rather than a resident application. If the locator is valid, the operation succeeds, but the broadcast application then receives a `SelectionFailedEvent`.

Likewise assume the platform supports multiple broadcast applications but not (for the present) a resident application. If the broadcast application selects a resident application, and if the locator is valid, the operation succeeds, but the broadcast application then receives a `SelectionFailedEvent`. (The second case is rather improbable. If the platform does not support resident applications, it is not clear how the broadcast application could obtain a valid locator to a broadcast application. The platform would have to support resident applications in concept, but not at the instant the broadcast applications selects a resident application.)

## Class Summary

### Interfaces

<b>Class Summary</b>	
<code>EmailClient</code>	Interface supporting the operations required on an email client.
<code>EmailClientService</code>	Service representing the resident email client
<code>InternetClient</code>	Base interface for the internet clients.
<code>InternetClientListener</code>	Interface for objects that wish to receive <code>InternetClientEvents</code>
<code>InternetClientService</code>	The base class for the interface to resident applications that are supported by the internet access profile.
<code>UsenetClient</code>	This interface supports the operations required on a Usenet news client.
<code>UsenetClientService</code>	Service representing the resident usenet news client
<code>WWWBrowser</code>	This interface provides support for the operations required on an WWW browser.
<code>WWWBrowserService</code>	Service representing a resident WWW browser
<b>Classes</b>	
<code>CancelledByUserEvent</code>	Event indicating that an operation on an internet client failed because the operation was canceled by the user
<code>HomePagePermission</code>	This class is for permissions related to the ability for an MHP application to set the home page of a WWW browser in the internet access profile.
<code>InternetClientEvent</code>	Base class for all status events from internet clients.
<code>InternetClientFailureEvent</code>	Event indicating that an operation on an internet client failed.
<code>InternetClientSuccessEvent</code>	Event indicating that an operation on an internet client succeeded
<code>InternetServiceFilter</code>	<code>InternetServiceFilter</code> represents a service type for a particular kind of internet client.
<code>InternetServiceType</code>	Class representing the additional service types available in the Internet Access profile.
<code>PermissionDeniedEvent</code>	Event indicating that an operation on an internet client failed due to the specified URL not being accessible by the client due to access controls on the server.
<code>URLUnavailableEvent</code>	Event indicating that an operation on an internet client failed due to the specified URL not being available.
<b>Exceptions</b>	
<code>BookmarkExistsException</code>	Exception generated when an application tries to add a bookmark that already exists
<code>ClientNotRunningException</code>	Exception thrown when a method of <code>InternetClient</code> or one of its subclasses is called while the client is not running, for instance if the client has been terminated by the user.
<code>EntryExistsException</code>	Exception generated when an application tries to add a bookmark or address book entry that already exists

# org.dvb.internet BookmarkExistsException

## Syntax

```
public class BookmarkExistsException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--org.dvb.internet.BookmarkExistsException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Exception generated when an application tries to add a bookmark that already exists

## Constructors

---

### BookmarkExistsException()

```
public BookmarkExistsException()
```

Construct a `BookmarkExistsException` with no detail message

---

### BookmarkExistsException(String)

```
public BookmarkExistsException(java.lang.String message)
```

Construct a `BookmarkExistsException` with the specified detail message

#### Parameters:

message - the detail message

# org.dvb.internet CancelledByUserEvent

## Syntax

```
public class CancelledByUserEvent extends InternetClientFailureEvent
```

```
java.lang.Object  
|  
+--java.util.EventObject  
|  
+--InternetClientEvent  
|  
+--InternetClientFailureEvent  
|  
+--org.dvb.internet.CancelledByUserEvent
```

## All Implemented Interfaces:

[java.io.Serializable](#)

## Description

Event indicating that an operation on an internet client failed because the operation was canceled by the user

## Constructors

---

### CancelledByUserEvent(Object, URL)

```
public CancelledByUserEvent(java.lang.Object source, java.net.URL url)
```

Construct a new `CancelledByUserEvent`

#### Parameters:

`source` - the source of the event

`url` - the URL to which the event relates.

# org.dvb.internet ClientNotRunningException

## Syntax

```
public class ClientNotRunningException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
   |
   +--java.lang.Exception
      |
      +--org.dvb.internet.ClientNotRunningException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Exception thrown when a method of `InternetClient` or one of its subclasses is called while the client is not running, for instance if the client has been terminated by the user.

## Constructors

---

### ClientNotRunningException()

```
public ClientNotRunningException()
```

Construct a `ClientNotRunningException` with no detail message

---

### ClientNotRunningException(String)

```
public ClientNotRunningException(java.lang.String reason)
```

Construct a `ClientNotRunningException` with the specified detail message

#### Parameters:

`reason` - the reason why the exception was thrown

# org.dvb.internet EmailClient

## Syntax

```
public interface EmailClient extends InternetClient
```

## All Superinterfaces:

`InternetClient`, `javax.tv.service.selection.ServiceContentHandler`

## Description

Interface supporting the operations required on an email client.

## Methods

---

### **createMessage(String, String, String, String)**

```
public void createMessage(java.lang.String to, java.lang.String subject,  
                           java.lang.String messageBody, java.lang.String sender)
```

Create a new email message. If any of the parameters are an empty string, the user will be prompted for the missing information. If the sender is null, then the default email address will be used. Null values for any of the other parameters will cause an exception to be thrown.

This is an asynchronous operation, whose success or failure will be indicated by an `InternetClientSuccessEvent` or `InternetClientFailureEvent` or one of their subclasses.

Applications can send email 'silently', without any user intervention as long as the sender email address that is used is not that of the user. Whenever email is to be sent with the users email address, the user must be prompted before the email can be sent.

#### **Parameters:**

`to` - the address to which the email should be sent.

`subject` - the subject for the email.

`messageBody` - the body of the message.

`sender` - the sender address to be used.

#### **Throws:**

`NullPointerException` - if any of the `to` address, `subject` or `message body` are null.

`ClientNotRunningException` - if the client is not currently running.

`IllegalArgumentException` - if either the destination address or the sender is an empty string.



# org.dvb.internet EmailClientService

## Syntax

```
public interface EmailClientService extends InternetClientService
```

## All Superinterfaces:

[InternetClientService](#), [javax.tv.service.Service](#)

## Description

Service representing the resident email client

## Methods

---

### addToAddressBook(String, String)

```
public void addToAddressBook(java.lang.String address, java.lang.String name)
```

Add an entry to the address book. As a side-effect an entry previously added by this method may be lost. Implementations may restrict the number of entries a single MHP application or source of applications may add.

#### Parameters:

`address` - the address to be added to the address book

`name` - the name that should be associated with that address

#### Throws:

[EntryExistsException](#) - if an entry with both the same name and the same address already exists in the address book.

[IllegalArgumentException](#) - if the string passed as the email address does not conform to the internet email address format

[IOException](#) - if no more address book entries can be added due to a lack of storage space or a limitation in the client

[IOException](#)

---

### getUserEmailAddress()

```
public java.lang.String getUserEmailAddress()
```

Get the email address of the user. The returned value shall be the same as that obtained by reading the value of the "User @" preference (see [org.dvb.user.GeneralPreferences](#) for details).

#### Returns:

the email address of the user, or null if no email address is set or the email address is not available to the client.

#### Throws:

[SecurityException](#) - if the caller does not have a [UserPreferencePermission](#) with the name "read".

# org.dvb.internet EntryExistsException

## Syntax

```
public class EntryExistsException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
   |
   +--java.lang.Exception
       |
       +--org.dvb.internet.EntryExistsException
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Exception generated when an application tries to add a bookmark or address book entry that already exists

## Constructors

---

### EntryExistsException()

```
public EntryExistsException()
```

Construct a `EntryExistsException` with no detail message

---

### EntryExistsException(String)

```
public EntryExistsException(java.lang.String message)
```

Construct a `EntryExistsException` with the specified detail message

#### Parameters:

`message` - the reason why the exception was thrown

# org.dvb.internet HomePagePermission

## Syntax

```
public class HomePagePermission extends java.security.BasicPermission
```

```
java.lang.Object
|
+--java.security.Permission
   |
   +--java.security.BasicPermission
      |
      +--org.dvb.internet.HomePagePermission
```

## All Implemented Interfaces:

java.security.Guard, java.io.Serializable

## Description

This class is for permissions related to the ability for an MHP application to set the home page of a WWW browser in the internet access profile. If an application has this permission then shall be able to set the home page.

## Constructors

---

### HomePagePermission(String)

```
public HomePagePermission(java.lang.String name)
```

Creates a new `HomePagePermission`. The name parameter is not used and must be set to empty string `""`. Implementations of this version of this specification shall ignore the name, but it could be taken into use in future versions of this specification.

#### Parameters:

name - the name of the `HomePagePermission`. Not used, shall be `""`.

---

### HomePagePermission(String, String)

```
public HomePagePermission(java.lang.String name, java.lang.String actions)
```

Creates a new `HomePagePermission`. The name and actions parameters are not used. Implementations of this version of this specification shall ignore the name and actions, but they could be taken into use in future versions of this specification. This constructor exists for use by the Policy object to instantiate new `Permission` objects.

#### Parameters:

name - the name of the `HomePagePermission`. Not used, shall be `""`.

actions - Not used, shall be null.

# org.dvb.internet InternetClient

## Syntax

```
public interface InternetClient extends javax.tv.service.selection.ServiceContentHandler
```

## All Known Subinterfaces:

`EmailClient`, `UsenetClient`, `WWWBrowser`

## All Superinterfaces:

`javax.tv.service.selection.ServiceContentHandler`

## Description

Base interface for the internet clients. Access to those methods common to all running instances of the client (e.g. operations on bookmark lists) are all carried out through the service objects associated with the `InternetClient` object in question. These are accessed using the `getService()` method.

## Methods

---

### `addInternetClientListener(InternetClientListener)`

```
public void addInternetClientListener(InternetClientListener l)
```

Add a listener for `InternetClientEvents`. If the listener is already registered, or the client is not running, then calls to this method have no effect.

#### Parameters:

`l` - the listener to be added.

---

### `getService()`

```
public InternetClientService getService()
```

Get the service object which matches this internet client. In the case of a web browser, for example, this would be an instance of the `WWWBrowserService` class.

#### Returns:

the service which matches the `InternetClient` object.

---

### `getServiceContentLocators()`

```
public javax.tv.locator.Locator[] getServiceContentLocators()
```

Reports the portions of the service on which this handler operates.

#### Overrides:

`getServiceContentLocators()` in interface `javax.tv.service.selection.ServiceContentHandler`

#### Returns:

An array of length 1, containing the locator representing the internet client. This shall be the same locator returned by calls to `getService().getLocator()`.

---

**removeInternetClientListener(InternetClientListener)**

```
public void removeInternetClientListener(InternetClientListener l)
```

Remove a listener for `InternetClientEvents`. If the listener is not registered, or the client is not running, then calls to this method have no effect.

**Parameters:**

l - the listener to be added.

# org.dvb.internet InternetClientEvent

## Syntax

```
public class InternetClientEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.dvb.internet.InternetClientEvent
```

## Direct Known Subclasses:

```
InternetClientFailureEvent, InternetClientSuccessEvent
```

## All Implemented Interfaces:

```
java.io.Serializable
```

## Description

Base class for all status events from internet clients.

## Constructors

---

### InternetClientEvent(Object, URL)

```
public InternetClientEvent(java.lang.Object source, java.net.URL url)
```

Construct a new `InternetClientEvent`

#### Parameters:

- `source` - the source of the event
- `url` - the URL to which the event relates.

## Methods

---

### getUrl()

```
public java.net.URL getUrl()
```

Get the URL for which this event was generated. In the case that a URL was not specified when the event was constructed, null shall be returned.

#### Returns:

an instance of `java.net.URL` or null

# org.dvb.internet InternetClientFailureEvent

## Syntax

```
public class InternetClientFailureEvent extends InternetClientEvent
```

```
java.lang.Object
|
+--java.util.EventObject
    |
    +--InternetClientEvent
        |
        +--org.dvb.internet.InternetClientFailureEvent
```

## Direct Known Subclasses:

CancelledByUserEvent, PermissionDeniedEvent, URLUnavailableEvent

## All Implemented Interfaces:

java.io.Serializable

## Description

Event indicating that an operation on an internet client failed. Typically, internet clients will post subclasses of this event detailing the reason why the operation failed.

## Constructors

---

### InternetClientFailureEvent(Object, URL)

```
public InternetClientFailureEvent(java.lang.Object source, java.net.URL url)
```

Construct a new InternetClientFailureEvent

#### Parameters:

`source` - the source of the event

`url` - the URL to which the event relates.

# org.dvb.internet InternetClientListener

## Syntax

```
public interface InternetClientListener extends java.util.EventListener
```

## All Superinterfaces:

java.util.EventListener

## Description

Interface for objects that wish to receive `InternetClientEvents`

## Methods

---

### ReceiveInternetClientEvent(InternetClientEvent)

```
public void ReceiveInternetClientEvent(InternetClientEvent event)
```

The method to be called when an `InternetClientEvent` is received.

#### Parameters:

`event` - the received `InternetClientEvent`



# org.dvb.internet InternetClientService

## Syntax

```
public interface InternetClientService extends javax.tv.service.Service
```

## All Known Subinterfaces:

`EmailClientService`, `UsenetClientService`, `WWWBrowserService`

## All Superinterfaces:

`javax.tv.service.Service`

## Description

The base class for the interface to resident applications that are supported by the internet access profile.

The lifecycle of an application which implements this interface or its subclasses is for a broadcast service. The application is started by selecting the appropriate service (using the `Locator` object returned by calls to `getLocator()` ). If this service is selected in the service context which contains the executing application, any currently presented content will be stopped and the application will be destroyed before the client is launched. Calling `destroy()` or `stop()` on the service context in which the client is running will cause the client to be terminated.

Methods in this API will not affect the lifecycle of the calling application.

## Methods

---

### canRunApplication()

```
public boolean canRunApplication()
```

Returns true if the application can run without having to stop the downloaded MHP application.

**Returns:**

true if the application can be run without stopping the calling application, or false otherwise.

---

### getName()

```
public java.lang.String getName()
```

Returns a short service name or an acronym. In the case of subclasses of `InternetClient`, the returned value is implementation dependent

**Overrides:**

`javax.tv.service.Service.getName()` in interface `javax.tv.service.Service`

**Returns:**

A string representing this service's short name.

---

### getServiceType()

```
public javax.tv.service.ServiceType getServiceType()
```

Returns the type of this service. In the case of internet clients, one of the service types defined in the `InternetServiceType` class shall be returned.

**Overrides:**

`javax.tv.service.Service.getServiceType()` in interface `javax.tv.service.Service`

**Returns:**

The service type of this Service.

---

**getSupportedClientServices()**

```
public InternetClientService[] getSupportedClientServices()
```

Returns all `InternetClientServices` supported by the same application as this one. This `InternetClientService` is included in the array.

**Returns:**

an array of `InternetClientServices`

---

**hasMultipleInstances()**

```
public boolean hasMultipleInstances()
```

This method indicates whether the service represented by this Service is available on multiple transports. This method has no effect in the case of an `InternetClient`

**Overrides:**

`javax.tv.service.Service.hasMultipleInstances()` in interface `javax.tv.service.Service`

**Returns:**

`FALSE` always for `InternetClient` instances

---

**retrieveDetails(SIRequestor)**

```
public javax.tv.service.SIRequest retrieveDetails(javax.tv.service.SIRequestor requestor)
```

This method will always fail when called for an `InternetClient`. The requestor will always be notified of a failure of type `DATA_UNAVAILABLE`.

**Overrides:**

`javax.tv.service.Service.retrieveDetails(javax.tv.service.SIRequestor)` in interface `javax.tv.service.Service`

**Parameters:**

`requestor` - - The `SIRequestor` to be notified when this retrieval operation completes.

**Returns:**

An `SIRequest` object identifying the request

# org.dvb.internet InternetClientSuccessEvent

## Syntax

```
public class InternetClientSuccessEvent extends InternetClientEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--InternetClientEvent
|
+--org.dvb.internet.InternetClientSuccessEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Event indicating that an operation on an internet client succeeded

## Constructors

---

### InternetClientSuccessEvent(Object, URL)

```
public InternetClientSuccessEvent(java.lang.Object source, java.net.URL url)
```

Construct a new `InternetClientSuccessEvent`

#### Parameters:

`source` - the source of the event

`url` - the URL to which the event relates.

# org.dvb.internet InternetServiceFilter

## Syntax

```
public final class InternetServiceFilter extends javax.tv.service.navigation.ServiceFilter
    java.lang.Object
    |
    +--javax.tv.service.navigation.ServiceFilter
    |
    +--org.dvb.internet.InternetServiceFilter
```

## Description

`InternetServiceFilter` represents a service type for a particular kind of internet client. A `ServiceList` resulting from this filter will include only services providing access to the specified type of internet client.

## Fields

---

### EMAIL\_CLIENT

```
public static final int EMAIL_CLIENT
Constant identifying an email client service
```

---

### NEWS\_CLIENT

```
public static final int NEWS_CLIENT
Constant identifying a usenet news client service
```

---

### WWW\_CLIENT

```
public static final int WWW_CLIENT
Constant identifying a WWW client service
```

## Constructors

---

### InternetServiceFilter(int)

```
public InternetServiceFilter(int service_type)
```

Constructs the filter based on a particular type of internet client service. The types of service required are those defined by the constants in this class. Support for other values is platform dependent. Platforms not supporting services of the type specified shall return an empty `ServiceList` when instances of this class constructed using that type are used.

#### Parameters:

`service_type` - the type of service required

## Methods

---

### **accept(Service)**

```
public boolean accept(javax.tv.service.Service service)
```

Tests if a particular service represents an internet client of the type specified in the constructor of this instance.

#### **Overrides:**

javax.tv.service.navigation.ServiceFilter.accept(javax.tv.service.Service) in class  
javax.tv.service.navigation.ServiceFilter

#### **Parameters:**

*service* - A Service to be evaluated against the filtering algorithm.

#### **Returns:**

true if service satisfies the filtering algorithm; false otherwise.

# org.dvb.internet InternetServiceType

## Syntax

```
public class InternetServiceType extends javax.tv.service.ServiceType
```

```
java.lang.Object
|
+--javax.tv.service.ServiceType
|
+--org.dvb.internet.InternetServiceType
```

## Description

Class representing the additional service types available in the Internet Access profile. When this service type is used in a `java.tv.service.navigation.ServiceTypeFilter` for filtering available services on their type, all internet client services shall be returned. Applications wishing to obtain a specific sub-type of internet client service should use `InternetServiceFilter`.

## See Also:

[InternetServiceFilter](#)

## Fields

---

### INTERNET\_CLIENT

```
public static final javax.tv.service.ServiceType INTERNET_CLIENT
WWW service type
```

## Constructors

---

### InternetServiceType(String)

```
protected InternetServiceType(java.lang.String name)
```

This protected constructor is provided for implementation use and to enable future evolution of this or other specifications. It shall not be called by inter-operable applications.

#### Parameters:

name - the name of the service type

# org.dvb.internet PermissionDeniedEvent

## Syntax

```
public class PermissionDeniedEvent extends InternetClientFailureEvent
```

```
java.lang.Object
|
+--java.util.EventObject
   |
   +--InternetClientEvent
      |
      +--InternetClientFailureEvent
         |
         +--org.dvb.internet.PermissionDeniedEvent
```

## All Implemented Interfaces:

[java.io.Serializable](#)

## Description

Event indicating that an operation on an internet client failed due to the specified URL not being accessible by the client due to access controls on the server.

## Constructors

---

### PermissionDeniedEvent(Object, URL)

```
public PermissionDeniedEvent(java.lang.Object source, java.net.URL url)
```

Construct a new `PermissionDeniedEvent`

#### Parameters:

`source` - the source of the event

`url` - the URL to which the event relates.

# org.dvb.internet URLUnavailableEvent

## Syntax

```
public class URLUnavailableEvent extends InternetClientFailureEvent
```

```
java.lang.Object
|
+--java.util.EventObject
    |
    +--InternetClientEvent
        |
        +--InternetClientFailureEvent
            |
            +--org.dvb.internet.URLUnavailableEvent
```

## All Implemented Interfaces:

java.io.Serializable

## Description

Event indicating that an operation on an internet client failed due to the specified URL not being available. This could be because a server is not available or because a file is not available on that server

## Constructors

---

### URLUnavailableEvent(Object, URL)

```
public URLUnavailableEvent(java.lang.Object source, java.net.URL url)
```

Construct a new URLUnavailableEvent

#### Parameters:

`source` - the source of the event

`url` - the URL to which the event relates.



# org.dvb.internet UsenetClient

## Syntax

```
public interface UsenetClient extends InternetClient
```

## All Superinterfaces:

[InternetClient](#), [javax.tv.service.selection.ServiceContentHandler](#)

## Description

This interface supports the operations required on a Usenet news client.

Any URLs passed to methods in this interface should correspond to the usenet news URL format specified in RFC 1738

## Methods

---

### selectGroup(URL)

```
public void selectGroup(java.net.URL group)
```

Select and display messages in the specified newsgroup. If the news client is not running, then this method will not cause it to be started and the call will fail.

This is an asynchronous operation, whose success or failure will be indicated by an [InternetClientSuccessEvent](#) or [InternetClientFailureEvent](#) or one of their subclasses.

#### Parameters:

`group` - the URL of the group. This may or may not include the address of a news server.

#### Throws:

[SecurityException](#) - if the caller does not have a [SocketPermission](#) for the host part of the specified URL.

[IllegalArgumentException](#) - if the specified URL does not correspond to the Usenet news URL format specified in RFC 1738

[ClientNotRunningException](#) - if the client is not currently running.

---

### selectMessage(URL)

```
public void selectMessage(java.net.URL message)
```

Select and display a message with the given message ID. If the news client is not running, then this method will not cause it to be started and the call will fail.

This is an asynchronous operation, whose success or failure will be indicated by an [InternetClientSuccessEvent](#) or [InternetClientFailureEvent](#) or one of their subclasses.

#### Parameters:

`message` - the URL of the message. This may or may not include the address of a news server.

#### Throws:

`SecurityException` - if the caller does not have a `SocketPermission` for the host part of the specified URL

`IllegalArgumentException` - if the specified URL does not include a Usenet news message ID or does not correspond to the Usenet news URL format specified in RFC 1738

`ClientNotRunningException` - if the client is not currently running.

# org.dvb.internet UsenetClientService

## Syntax

```
public interface UsenetClientService extends InternetClientService
```

## All Superinterfaces:

[InternetClientService](#), [javax.tv.service.Service](#)

## Description

Service representing the resident usenet news client

## Methods

---

### subscribe(String)

```
public void subscribe(java.lang.String newsgroup)
```

Add a newsgroup to the list currently subscribed newsgroups. If the newsgroup is already in the list, then this method has no effect. As a side-effect a newsgroup previously subscribed to by this method may be unsubscribed. Implementations may restrict the number of newsgroups a single MHP application or source of applications may subscribe to.

#### Parameters:

`newsgroup` - the name of the newsgroup that should be subscribed to

#### Throws:

`IOException` - if no more newsgroups can be added due to a lack of storage space

`IOException`

# org.dvb.internet WWWBrowser

## Syntax

```
public interface WWWBrowser extends InternetClient
```

## All Superinterfaces:

`InternetClient`, `javax.tv.service.selection.ServiceContentHandler`

## Description

This interface provides support for the operations required on an WWW browser.

Any URLs passed to methods in this interface should correspond to the HTTP or FTP URL schemes specified in RFC 1738. Other schemes or URL formats may be supported, but these are implementation-specific and additional supported schemes should be discovered by querying the capabilities of the web browser if they are required.

## Methods

---

### goToURL(URL)

```
public void goToURL(java.net.URL url)
```

Direct the web browser to display the page at the specified URL. If the web browser is not already running, then this method will not cause the web browser to be started and the call will fail.

This is an asynchronous operation, whose success or failure will be indicated by an `InternetClientSuccessEvent` or `InternetClientFailureEvent` or one of their subclasses.

#### Parameters:

`url` - the URL to visit

#### Throws:

`SecurityException` - if the caller does not have a `SocketPermission` for the host part of the specified URL

`IllegalArgumentException` - if the URL scheme is not supported by the web browser.

`ClientNotRunningException` - if the client is not currently running.

# org.dvb.internet WWWBrowserService

## Syntax

```
public interface WWWBrowserService extends InternetClientService
```

## All Superinterfaces:

[InternetClientService](#), [javax.tv.service.Service](#)

## Description

Service representing a resident WWW browser

## Methods

---

### addBookmark(Locator, String)

```
public void addBookmark(javax.tv.locator.Locator locator, java.lang.String name)
```

Add a bookmark to the list of bookmarks in the current application. As a side-effect a bookmark previously added by this method may be lost. Implementations may restrict the number of bookmarks a single MHP application or source of applications may add.

#### Parameters:

`locator` - the Locator that should be added to the bookmarks list

`name` - the name that should be displayed for that locator in the bookmarks list. Multiple entries in the bookmarks list may not have the same name.

#### Throws:

[EntryExistsException](#) - if a bookmark with both the same name and the same locator already exists in the bookmarks list.

[IOException](#) - if no more bookmarks can be added due to a lack of storage space or a limitation in the client

[IOException](#)

---

### addBookmark(URL, String)

```
public void addBookmark(java.net.URL bookmarkUrl, java.lang.String name)
```

Add a bookmark to the list of bookmarks in the current application. As a side-effect a bookmark previously added by this method may be lost. Implementations may restrict the number of bookmarks a single MHP application or source of applications may add.

#### Parameters:

`bookmarkUrl` - the URL that should be added to the bookmarks list.

`name` - the name that should be displayed for that URL in the bookmarks list.

#### Throws:

[EntryExistsException](#) - if a bookmark with both the same name and the same URL already exists in the bookmarks list.

`java.lang.IllegalArgumentException` - if the URL scheme is not supported by the application.

`IOException` - if no more bookmarks can be added due to a lack of storage space or other limitation in the client

`IOException`

---

### **areFramesSupported()**

```
public boolean areFramesSupported()
```

Check whether frames are supported by the browser and enabled.

**Returns:**

true if the browser supports frames and frame support is enabled by the user, false otherwise.

---

### **getAcceptedMediaTypes()**

```
public java.lang.String[] getAcceptedMediaTypes()
```

Returns an array of supported MIME types, e.g. "application/vnd.rn-realmedia" The returned MIME types shall include at least one entry with type "text/html". Each entry of this type shall have a 'version' parameter that indicates the version of HTML that is supported, for example text/html; version = "4.0". A browser may claim to support an HTML version while only implementing a subset of the features - this is implementation-dependent. A browser that wishes to explicitly indicate support for multiple HTML versions shall return multiple entries of type "text/html" with different values for the 'version' parameter.

**Returns:**

an array of MIME types supported by the web browser

---

### **getSupportedPlugins()**

```
public java.lang.String[] getSupportedPlugins()
```

Returns an array of names of installed plug-ins. The name in each string is defined by the plug-in provider

**Returns:**

an array of plugin names

---

### **getUserAgent()**

```
public java.lang.String getUserAgent()
```

Returns the string used in the HTTP "User-Agent" header.

**Returns:**

the string identifying the user agent

---

### **setHomepage(URL)**

```
public void setHomepage(java.net.URL defaultUrl)
```

Set the home page for the web browser.

**Parameters:**

`defaultUrl` - the URL to be used as the default when the application is launched with no starting URL.

**Throws:**

`SecurityException` - if the caller does not have a `HomePagePermission`

`IllegalArgumentException` - if the URL scheme is not supported by the application.

Different classes which implement or extend this interface may implement different schemes.

---

## **Annex AI (normative): Non-CA Smartcard APIs**



# Package

# org.dvb.smartcard

## Description

Provides DVB specific support for smart card reader APIs.

### Class Summary

#### Classes

`SmartCardPermission`

This class is for permissions related to access to smart cards via the smart card API.

# org.dvb.smartcard SmartCardPermission

## Syntax

```
public class SmartCardPermission extends java.security.BasicPermission
```

```
java.lang.Object
|
+--java.security.Permission
   |
   +--java.security.BasicPermission
      |
      +--org.dvb.smartcard.SmartCardPermission
```

## All Implemented Interfaces:

java.security.Guard, java.io.Serializable

## Description

This class is for permissions related to access to smart cards via the smart card API. A SmartCardPermission contains no name and no actions list. If an application has the SmartCard permission, then it is allowed to access the SmartCard API and any smart cards via that API.

## Constructors

---

### SmartCardPermission(String)

```
public SmartCardPermission(java.lang.String name)
```

Creates a new SmartCardPermission. The name parameter is not used and must be set to empty string "". Implementations of this version of this specification shall ignore the name, but it could be taken into use in future versions of this specification.

#### Parameters:

`name` - the name of the SmartCardPermission. Not used, shall be "".

---

### SmartCardPermission(String, String)

```
public SmartCardPermission(java.lang.String name, java.lang.String actions)
```

Creates a new SmartCardPermission. The name and actions parameters are not used. Implementations of this version of this specification shall ignore the name and actions, but they could be taken into use in future versions of this specification. This constructor exists for use by the Policy object to instantiate new Permission objects.

#### Parameters:

`name` - the name of the SmartCardPermission. Not used, shall be "".

`actions` - Not used, shall be null.

# Index

## Symbols

- a - of org.havi.ui.HEventMulticaster 866
- abort() - of org.dvb.dsmcc.DSMCCObject 632
- accept(AppID) - of org.dvb.application.AppsDatabaseFilter 735
- accept(AppID) - of org.dvb.application.CurrentServiceFilter 739
- accept(Service) - of org.dvb.internet.InternetServiceFilter 1385
- ACTIONED\_FOCUSED\_STATE - of org.havi.ui.HState 1088
- ACTIONED\_STATE - of org.havi.ui.HState 1088
- ACTIONED\_STATE\_BIT - of org.havi.ui.HState 1088
- actionPerformed(ActionEvent) - of org.havi.ui.HEventMulticaster 866
- ActiveFormatDescriptionChangedEvent - of org.dvb.media 577
- ActiveFormatDescriptionChangedEvent(Object, int) - of org.dvb.media.ActiveFormatDescriptionChangedEvent 577
- add(HActionListener, HActionListener) - of org.havi.ui.HEventMulticaster 867
- add(HAdjustmentListener, HAdjustmentListener) - of org.havi.ui.HEventMulticaster 867
- add(HBackgroundImageListener, HBackgroundImageListener) - of org.havi.ui.HEventMulticaster 867
- add(HFocusListener, HFocusListener) - of org.havi.ui.HEventMulticaster 867
- add(HItemListener, HItemListener) - of org.havi.ui.HEventMulticaster 867
- add(HKeyListener, HKeyListener) - of org.havi.ui.HEventMulticaster 867
- add(HScreenConfigurationListener, HScreenConfigurationListener) - of org.havi.ui.HEventMulticaster 868
- add(HScreenConfigurationListener, HScreenConfigurationListener, HScreenConfigTemplate) - of org.havi.ui.HEventMulticaster 868
- add(HScreenLocationModifiedListener, HScreenLocationModifiedListener) - of org.havi.ui.HEventMulticaster 868
- add(HTextListener, HTextListener) - of org.havi.ui.HEventMulticaster 868
- add(HToggleButton) - of org.havi.ui.HToggleGroup 1154
- add(int, String) - of org.dvb.user.Preference 482
- add(ResourceStatusListener, ResourceStatusListener) - of org.havi.ui.HEventMulticaster 868
- add(String) - of org.dvb.user.Preference 482
- add(String[]) - of org.dvb.user.Preference 482
- add(WindowListener, WindowListener) - of org.havi.ui.HEventMulticaster 869
- ADD\_INDEX\_END - of org.havi.ui.HListGroup 948
- addAdjustmentListener(HAdjustmentListener) - of org.havi.ui.HAdjustmentValue 811
- addAdjustmentListener(HAdjustmentListener) - of org.havi.ui.HRangeValue 1013
- addAfter(Component, Component) - of org.havi.ui.HComponentOrdering 846
- addAfter(Component, Component) - of org.havi.ui.HContainer 853
- addAfter(Component, Component) - of org.havi.ui.HScene 1023
- addAllArrowKeys() - of org.dvb.event.UserEventRepository 467
- addAllColourKeys() - of org.dvb.event.UserEventRepository 467
- addAllNumericKeys() - of org.dvb.event.UserEventRepository 467
- addAppStateChangeListener(AppStateChangeListener) - of org.dvb.application.AppProxy 722
- addBefore(Component, Component) - of org.havi.ui.HComponentOrdering 846
- addBefore(Component, Component) - of org.havi.ui.HContainer 854
- addBefore(Component, Component) - of org.havi.ui.HScene 1024
- addBookmark(Locator, String) - of org.dvb.internet.WWWBrowserService 1393
- addBookmark(URL, String) - of org.dvb.internet.WWWBrowserService 1393
- addBouquetMonitoringListener(SIMonitoringListener, int) - of org.dvb.si.SIDatabase 509
- addConnectionListener(ConnectionListener) - of org.dvb.net.rc.ConnectionRCInterface 696
- addEventPresentFollowingMonitoringListener(SIMonitoringListener, int, int, int) - of org.dvb.si.SIDatabase 509
- addEventScheduleMonitoringListener(SIMonitoringListener, int, int, int, Date, Date) - of org.dvb.si.SIData-

base 510

addExclusiveAccessToAWTEvent(ResourceClient, UserEventRepository) - of org.dvb.event.EventManager 457

addHActionListener(HActionListener) - of org.havi.ui.HActionable 802

addHActionListener(HActionListener) - of org.havi.ui.HGraphicButton 898

addHActionListener(HActionListener) - of org.havi.ui.HTextButton 1126

addHActionListener(HActionListener) - of org.havi.ui.HToggleButton 1146

addInnerApplicationListener(InnerApplicationListener) - of org.dvb.application.inner.InnerApplicationContainer 1322

addInternal(EventListener, EventListener) - of org.havi.ui.HEventMulticaster 869

addInternetClientListener(InternetClientListener) - of org.dvb.internet.InternetClient 1376

addItem(HListElement, int) - of org.havi.ui.HListGroup 949

addItemListener(HItemListener) - of org.havi.ui.HItemValue 939

addItemListener(HItemListener) - of org.havi.ui.HListGroup 950

addItems(HListElement[], int) - of org.havi.ui.HListGroup 950

addKey(int) - of org.dvb.event.UserEventRepository 467

addListener(AppsDatabaseEventListener) - of org.dvb.application.AppsDatabase 728

addNetworkMonitoringListener(SIMonitoringListener, int) - of org.dvb.si.SIDatabase 511

addNPTListener(NPTListener) - of org.dvb.dsmcc.DSMCCStream 638

addObjectChangeEventListener(ObjectChangeEventListener) - of org.dvb.dsmcc.DSMCCObject 633

addOnScreenLocationModifiedListener(HScreenLocationModifiedListener) - of org.havi.ui.HVideoComponent 1162

addPMTServiceMonitoringListener(SIMonitoringListener, int, int, int) - of org.dvb.si.SIDatabase 511

addResourceStatusEventListener(ResourceStatusListener) - of org.dvb.event.EventManager 457

addResourceStatusEventListener(ResourceStatusListener) - of org.dvb.net.rc.RCInterfaceManager 705

addResourceStatusEventListener(ResourceStatusListener) - of org.havi.ui.HScreenDevice 1055

addScreenConfigurationListener(HScreenConfigurationListener) - of org.havi.ui.HScreenDevice 1055

addScreenConfigurationListener(HScreenConfigurationListener, HScreenConfigTemplate) - of org.havi.ui.HScreenDevice 1055

addServiceMonitoringListener(SIMonitoringListener, int, int) - of org.dvb.si.SIDatabase 512

addShortcut(int, HActionable) - of org.havi.ui.HScene 1024

addSubtitleListener(SubtitleListener) - of org.dvb.media.SubtitlingEventControl 601

addTextOverflowListener(TextOverflowListener) - of org.dvb.ui.DVBTextLayoutManager 778

addToAddressBook(String, String) - of org.dvb.internet.EmailClientService 1373

addUserEvent(UserEvent) - of org.dvb.event.UserEventRepository 467

addUserEventListener(UserEventListener, ResourceClient, UserEventRepository) - of org.dvb.event.EventManager 458

addUserEventListener(UserEventListener, UserEventRepository) - of org.dvb.event.EventManager 458

addUserPreferenceChangeListener(UserPreferenceChangeListener) - of org.dvb.user.UserPreferenceManager 487

addVideoFormatListener(VideoFormatListener) - of org.dvb.media.VideoFormatControl 605

addWindowListener(WindowListener) - of org.dvb.application.inner.DVBScene 1316

addWindowListener(WindowListener) - of org.havi.ui.HScene 1025

ADJUST\_BUTTON\_LESS - of org.havi.ui.HAdjustableLook 806

ADJUST\_BUTTON\_MORE - of org.havi.ui.HAdjustableLook 806

ADJUST\_END\_CHANGE - of org.havi.ui.event.HAdjustmentEvent 1199

ADJUST\_FIRST - of org.havi.ui.event.HAdjustmentEvent 1200

ADJUST\_LAST - of org.havi.ui.event.HAdjustmentEvent 1200

ADJUST\_LESS - of org.havi.ui.event.HAdjustmentEvent 1200

ADJUST\_MORE - of org.havi.ui.event.HAdjustmentEvent 1200

ADJUST\_NONE - of org.havi.ui.HAdjustableLook 806

ADJUST\_PAGE\_LESS - of org.havi.ui.event.HAdjustmentEvent 1200

ADJUST\_PAGE\_LESS - of org.havi.ui.HAdjustableLook 806

ADJUST\_PAGE\_MORE - of org.havi.ui.event.HAdjustmentEvent 1200

ADJUST\_PAGE\_MORE - of org.havi.ui.HAdjustableLook 806

ADJUST\_START\_CHANGE - of org.havi.ui.event.HAdjustmentEvent 1201  
ADJUST\_THUMB - of org.havi.ui.HAdjustableLook 807  
ADJUSTMENT\_VALUE\_CHANGE - of org.havi.ui.HVisible 1179  
AFD\_14\_9 - of org.dvb.media.VideoFormatControl 602  
AFD\_14\_9\_TOP - of org.dvb.media.VideoFormatControl 602  
AFD\_16\_9 - of org.dvb.media.VideoFormatControl 602  
AFD\_16\_9\_SP\_14\_9 - of org.dvb.media.VideoFormatControl 602  
AFD\_16\_9\_SP\_4\_3 - of org.dvb.media.VideoFormatControl 602  
AFD\_16\_9\_TOP - of org.dvb.media.VideoFormatControl 603  
AFD\_4\_3 - of org.dvb.media.VideoFormatControl 603  
AFD\_4\_3\_SP\_14\_9 - of org.dvb.media.VideoFormatControl 603  
AFD\_GT\_16\_9 - of org.dvb.media.VideoFormatControl 603  
AFD\_NOT\_PRESENT - of org.dvb.media.VideoFormatControl 603  
AFD\_SAME - of org.dvb.media.VideoFormatControl 603  
ALL\_STATES - of org.havi.ui.HState 1088  
ALPHABETIC\_PRESENTATION\_FORMS\_A - of org.havi.ui.HFontCapabilities 884  
ANIMATE\_CONTENT\_CHANGE - of org.havi.ui.HVisible 1179  
APP\_ADDED - of org.dvb.application.AppsDatabaseEvent 731  
APP\_CHANGED - of org.dvb.application.AppsDatabaseEvent 731  
APP\_DELETED - of org.dvb.application.AppsDatabaseEvent 731  
AppAttributes - of org.dvb.application 713  
AppIcon - of org.dvb.application 718  
AppIcon() - of org.dvb.application.AppIcon 718  
AppID - of org.dvb.application 719  
AppID(int, int) - of org.dvb.application.AppID 719  
ApplicationDownloadException - of org.dvb.application.storage 1349  
ApplicationDownloadException() - of org.dvb.application.storage.ApplicationDownloadException 1349  
ApplicationDownloadException(String) - of org.dvb.application.storage.ApplicationDownloadException 1349  
ApplicationSecurityContext - of org.dvb.application.plugins 1330  
ApplicationSecurityContext(URL[], Locator) - of org.dvb.application.plugins.ApplicationSecurityContext 1330  
ApplicationStorage - of org.dvb.application.storage 1350  
ApplicationStoragePermission - of org.dvb.application.storage 1353  
ApplicationStoragePermission(String) - of org.dvb.application.storage.ApplicationStoragePermission 1353  
ApplicationStoragePermission(String, String) - of org.dvb.application.storage.ApplicationStoragePermission 1354  
AppProxy - of org.dvb.application 721  
AppsControlPermission - of org.dvb.application 725  
AppsControlPermission() - of org.dvb.application.AppsControlPermission 725  
AppsControlPermission(String, String) - of org.dvb.application.AppsControlPermission 725  
AppsDatabase - of org.dvb.application 727  
AppsDatabaseEvent - of org.dvb.application 731  
AppsDatabaseEvent(int, AppID, Object) - of org.dvb.application.AppsDatabaseEvent 732  
AppsDatabaseEventListener - of org.dvb.application 733  
AppsDatabaseFilter - of org.dvb.application 735  
AppsDatabaseFilter() - of org.dvb.application.AppsDatabaseFilter 735  
AppStateChangeEvent - of org.dvb.application 736  
AppStateChangeEvent(AppID, int, int, Object, boolean) - of org.dvb.application.AppStateChangeEvent 736  
AppStateChangeEventListener - of org.dvb.application 738  
ARABIC\_EXTENDED - of org.havi.ui.HFontCapabilities 884  
ARABIC\_PRESENTATION\_FORMS\_A - of org.havi.ui.HFontCapabilities 884  
ARABIC\_PRESENTATION\_FORMS\_B - of org.havi.ui.HFontCapabilities 884  
areFramesSupported() - of org.dvb.internet.WWWBrowserService 1394  
ARMENIAN - of org.havi.ui.HFontCapabilities 884

ARROWS - of org.havi.ui.HFontCapabilities 885  
ASPECT\_RATIO\_16\_9 - of org.dvb.media.VideoFormatControl 603  
ASPECT\_RATIO\_2\_21\_1 - of org.dvb.media.VideoFormatControl 603  
ASPECT\_RATIO\_4\_3 - of org.dvb.media.VideoFormatControl 603  
ASPECT\_RATIO\_UNKNOWN - of org.dvb.media.VideoFormatControl 603  
AspectRatioChangedEvent - of org.dvb.media 578  
AspectRatioChangedEvent(Object, int) - of org.dvb.media.AspectRatioChangedEvent 578  
asynchronousLoad(AsynchronousLoadingEventListener) - of org.dvb.dsmcc.DSMCCObject 633  
AsynchronousLoadingEvent - of org.dvb.dsmcc 628  
AsynchronousLoadingEvent(DSMCCObject) - of org.dvb.dsmcc.AsynchronousLoadingEvent 628  
AsynchronousLoadingEventListener - of org.dvb.dsmcc 629  
attach(byte[]) - of org.dvb.dsmcc.ServiceDomain 672  
attach(Locator) - of org.dvb.dsmcc.ServiceDomain 672  
attach(Locator, int) - of org.dvb.dsmcc.ServiceDomain 672  
b - of org.havi.ui.HEventMulticaster 866  
BACKGROUND\_FILL - of org.havi.ui.HScene 1022  
BACKGROUND\_FILL - of org.havi.ui.HVisible 1179  
BACKGROUNDIMAGE\_FILE\_NOT\_FOUND - of org.havi.ui.event.HBackgroundImageEvent 1203  
BACKGROUNDIMAGE\_FIRST - of org.havi.ui.event.HBackgroundImageEvent 1203  
BACKGROUNDIMAGE\_INVALID - of org.havi.ui.event.HBackgroundImageEvent 1203  
BACKGROUNDIMAGE\_IOERROR - of org.havi.ui.event.HBackgroundImageEvent 1204  
BACKGROUNDIMAGE\_LAST - of org.havi.ui.event.HBackgroundImageEvent 1204  
BACKGROUNDIMAGE\_LOADED - of org.havi.ui.event.HBackgroundImageEvent 1204  
BackgroundVideoPresentationControl - of org.dvb.media 579  
BASIC\_ARABIC - of org.havi.ui.HFontCapabilities 885  
BASIC\_GEORGIAN - of org.havi.ui.HFontCapabilities 885  
BASIC\_GREEK - of org.havi.ui.HFontCapabilities 885  
BASIC\_HEBREW - of org.havi.ui.HFontCapabilities 885  
BASIC\_LATIN - of org.havi.ui.HFontCapabilities 885  
BENGALI - of org.havi.ui.HFontCapabilities 885  
bind(XletContext, String, Remote) - of org.dvb.io.ixc.IxcRegistry 1263  
bind(XletContext, String, Remote, boolean) - of org.dvb.io.ixc.IxcRegistry 1264  
BLOCK\_ELEMENTS - of org.havi.ui.HFontCapabilities 885  
BookmarkExistsException - of org.dvb.internet 1369  
BookmarkExistsException() - of org.dvb.internet.BookmarkExistsException 1369  
BookmarkExistsException(String) - of org.dvb.internet.BookmarkExistsException 1369  
BOPOMOFO - of org.havi.ui.HFontCapabilities 886  
BOUQUET - of org.dvb.si.SIMonitoringType 542  
BOUQUET\_NAME - of org.dvb.si.DescriptorTag 496  
BOX\_DRAWING - of org.havi.ui.HFontCapabilities 886  
brighter() - of org.dvb.ui.DVBColor 768  
BUSINESS\_GRAPHICS - of org.havi.ui.HImageHints 933  
CA\_FAILURE - of org.dvb.media.PresentationChangedEvent 590  
CA\_IDENTIFIER - of org.dvb.si.DescriptorTag 496  
CA\_RETURNED - of org.dvb.media.PresentationChangedEvent 590  
CABLE\_DELIVERY\_SYSTEM - of org.dvb.si.DescriptorTag 496  
CAException - of org.dvb.media 580  
CAException() - of org.dvb.media.CAException 580  
CAException(String) - of org.dvb.media.CAException 580  
CancelledByUserEvent - of org.dvb.internet 1370  
CancelledByUserEvent(Object, URL) - of org.dvb.internet.CancelledByUserEvent 1370  
cancelRequest() - of org.dvb.si.SIRequest 549  
canRunApplication() - of org.dvb.internet.InternetClientService 1381  
CAPermission - of org.dvb.net.ca 748  
CAPermission(String) - of org.dvb.net.ca.CAPermission 748

CAPermission(String, String) - of org.dvb.net.ca.CAPermission 749  
 CARET\_NEXT\_CHAR - of org.havi.ui.event.HTextEvent 1245  
 CARET\_NEXT\_LINE - of org.havi.ui.event.HTextEvent 1246  
 CARET\_NEXT\_PAGE - of org.havi.ui.event.HTextEvent 1246  
 CARET\_POSITION\_CHANGE - of org.havi.ui.HVisible 1179  
 CARET\_PREV\_CHAR - of org.havi.ui.event.HTextEvent 1246  
 CARET\_PREV\_LINE - of org.havi.ui.event.HTextEvent 1246  
 CARET\_PREV\_PAGE - of org.havi.ui.event.HTextEvent 1246  
 caretMoved(HTextEvent) - of org.havi.ui.event.HTextListener 1249  
 caretMoved(HTextEvent) - of org.havi.ui.HEventMulticaster 869  
 caretNextCharacter() - of org.havi.ui.HSinglelineEntry 1070  
 caretNextLine() - of org.havi.ui.HMultilineEntry 980  
 caretPreviousCharacter() - of org.havi.ui.HSinglelineEntry 1070  
 caretPreviousLine() - of org.havi.ui.HMultilineEntry 980  
 CARTOON - of org.havi.ui.HImageHints 933  
 CAsStopEvent - of org.dvb.media 581  
 CAsStopEvent(Controller) - of org.dvb.media.CAsStopEvent 581  
 CAsStopEvent(Controller, MediaLocator) - of org.dvb.media.CAsStopEvent 581  
 CHANGEABLE\_SINGLE\_COLOR - of org.havi.ui.HBackgroundConfigTemplate 831  
 checkPermission(Permission) - of org.dvb.application.plugins.ApplicationSecurityContext 1330  
 CJK\_COMPATIBILITY - of org.havi.ui.HFontCapabilities 886  
 CJK\_COMPATIBILITY\_FORMS - of org.havi.ui.HFontCapabilities 886  
 CJK\_COMPATIBILITY\_IDEOGRAPHS - of org.havi.ui.HFontCapabilities 886  
 CJK\_MISCELLANEOUS - of org.havi.ui.HFontCapabilities 886  
 CJK\_SYMBOLS\_AND\_PUNCTUATION - of org.havi.ui.HFontCapabilities 886  
 CJK\_UNIFIED\_IDEOGRAPHS - of org.havi.ui.HFontCapabilities 886  
 CLEAR - of org.dvb.ui.DVBAlphaComposite 756  
 Clear - of org.dvb.ui.DVBAlphaComposite 755  
 clearSelection() - of org.havi.ui.HListGroup 950  
 ClientNotRunningException - of org.dvb.internet 1371  
 ClientNotRunningException() - of org.dvb.internet.ClientNotRunningException 1371  
 ClientNotRunningException(String) - of org.dvb.internet.ClientNotRunningException 1371  
 close() - of org.davic.net.ca.MMIOObject 384  
 COMBINING\_DIACRITICAL\_MARKS - of org.havi.ui.HFontCapabilities 887  
 COMBINING\_DIACRITICAL\_MARKS\_FOR\_SYMBOLS - of org.havi.ui.HFontCapabilities 887  
 COMBINING\_HALF\_MARKS - of org.havi.ui.HFontCapabilities 887  
 COMPONENT - of org.dvb.si.DescriptorTag 496  
 connect() - of org.dvb.media.DripFeedDataSource 585  
 connect() - of org.dvb.net.rc.ConnectionRCInterface 696  
 connectionChanged(ConnectionRCEvent) - of org.dvb.net.rc.ConnectionListener 692  
 ConnectionEstablishedEvent - of org.dvb.net.rc 690  
 ConnectionEstablishedEvent(Object) - of org.dvb.net.rc.ConnectionEstablishedEvent 690  
 ConnectionFailedEvent - of org.dvb.net.rc 691  
 ConnectionFailedEvent(Object) - of org.dvb.net.rc.ConnectionFailedEvent 691  
 ConnectionListener - of org.dvb.net.rc 692  
 ConnectionParameters - of org.dvb.net.rc 693  
 ConnectionParameters(String, String, String) - of org.dvb.net.rc.ConnectionParameters 693  
 ConnectionParameters(String, String, String, InetAddress[]) - of org.dvb.net.rc.ConnectionParameters 693  
 ConnectionRCEvent - of org.dvb.net.rc 695  
 ConnectionRCEvent(Object) - of org.dvb.net.rc.ConnectionRCEvent 695  
 ConnectionRCInterface - of org.dvb.net.rc 696  
 ConnectionTerminatedEvent - of org.dvb.net.rc 700  
 ConnectionTerminatedEvent(Object) - of org.dvb.net.rc.ConnectionTerminatedEvent 700  
 CONTENT - of org.dvb.si.DescriptorTag 496  
 CONTENT\_CHANGE - of org.havi.ui.HVisible 1180

CONTROL\_PICTURES - of org.havi.ui.HFontCapabilities 887  
convertSISStringToJavaString(byte[], int, int, boolean) - of org.dvb.si.SIUtil 572  
convertTo(HScreenConfiguration, Point) - of org.havi.ui.HScreenConfiguration 1052  
COUNTRY\_AVAILABILITY - of org.dvb.si.DescriptorTag 496  
createEmbeddedContext(URL[], Locator) - of org.dvb.application.plugins.ApplicationSecurityContext 1331  
createFont(String, int, int) - of org.dvb.ui.FontFactory 784  
createGraphics() - of org.dvb.ui.DVBBufferedImage 761  
createMessage(String, String, String, String) - of org.dvb.internet.EmailClient 1372  
createStoredApplicationService(int, int, String) - of org.dvb.application.storage.StoredServiceCreator 1363  
CURRENCY\_SYMBOLS - of org.havi.ui.HFontCapabilities 887  
currentItemChanged(HItemEvent) - of org.havi.ui.event.HItemListener 1220  
currentItemChanged(HItemEvent) - of org.havi.ui.HEventMulticaster 869  
CurrentServiceFilter - of org.dvb.application 739  
CurrentServiceFilter() - of org.dvb.application.CurrentServiceFilter 739  
CYRILLIC - of org.havi.ui.HFontCapabilities 887  
D\_D2\_MAC - of org.dvb.si.SIServiceType 561  
DAR\_16\_9 - of org.dvb.media.VideoFormatControl 604  
DAR\_4\_3 - of org.dvb.media.VideoFormatControl 604  
darker() - of org.dvb.ui.DVBColor 769  
data - of org.havi.ui.HChangeData 842  
DATA\_BROADCAST - of org.dvb.si.DescriptorTag 497  
DATA\_BROADCAST - of org.dvb.si.SIServiceType 561  
DatagramSocketBufferControl - of org.dvb.net 687  
DEFAULT\_ICON\_HEIGHT - of org.havi.ui.HListGroup 948  
DEFAULT\_ICON\_WIDTH - of org.havi.ui.HListGroup 948  
DEFAULT\_LABEL\_HEIGHT - of org.havi.ui.HListGroup 948  
DEFAULT\_LABEL\_WIDTH - of org.havi.ui.HListGroup 949  
deleteNextChar() - of org.havi.ui.HSinglelineEntry 1070  
deletePreviousChar() - of org.havi.ui.HSinglelineEntry 1071  
Descriptor - of org.dvb.si 494  
DescriptorTag - of org.dvb.si 496  
DESTROYED - of org.dvb.application.AppProxy 721  
detach() - of org.dvb.dsmcc.ServiceDomain 673  
DEVANAGARI - of org.havi.ui.HFontCapabilities 887  
DFC\_PLATFORM - of org.dvb.media.VideoFormatControl 604  
DFC\_PROCESSING\_CCO - of org.dvb.media.VideoFormatControl 604  
DFC\_PROCESSING\_FULL - of org.dvb.media.VideoFormatControl 604  
DFC\_PROCESSING\_LB\_14\_9 - of org.dvb.media.VideoFormatControl 604  
DFC\_PROCESSING\_LB\_16\_9 - of org.dvb.media.VideoFormatControl 604  
DFC\_PROCESSING\_LB\_2\_21\_1\_ON\_16\_9 - of org.dvb.media.VideoFormatControl 604  
DFC\_PROCESSING\_LB\_2\_21\_1\_ON\_4\_3 - of org.dvb.media.VideoFormatControl 605  
DFC\_PROCESSING\_NONE - of org.dvb.media.VideoFormatControl 605  
DFC\_PROCESSING\_PAN\_SCAN - of org.dvb.media.VideoFormatControl 605  
DFC\_PROCESSING\_UNKNOWN - of org.dvb.media.VideoFormatControl 605  
DFCChangedEvent - of org.dvb.media 583  
DFCChangedEvent(Object, int) - of org.dvb.media.DFCChangedEvent 583  
DIGITAL\_RADIO\_SOUND - of org.dvb.si.SIServiceType 561  
DIGITAL\_TELEVISION - of org.dvb.si.SIServiceType 561  
DINGBATS - of org.havi.ui.HFontCapabilities 887  
DISABLED\_ACTIONED\_FOCUSED\_STATE - of org.havi.ui.HState 1089  
DISABLED\_ACTIONED\_STATE - of org.havi.ui.HState 1089  
DISABLED\_FOCUSED\_STATE - of org.havi.ui.HState 1089  
DISABLED\_STATE - of org.havi.ui.HState 1089  
DISABLED\_STATE\_BIT - of org.havi.ui.HState 1089  
disconnect() - of org.dvb.media.DripFeedDataSource 585



disconnect() - of org.dvb.net.rc.ConnectionRCInterface 696  
 displayImage(HBackgroundImage) - of org.havi.ui.HStillImageBackgroundConfiguration 1111  
 displayImage(HBackgroundImage, HScreenRectangle) - of org.havi.ui.HStillImageBackgroundConfiguration 1111  
 dispose() - of org.dvb.ui.DVBBufferedImage 761  
 dispose() - of org.havi.ui.HScene 1025  
 dispose() - of org.havi.ui.HSound 1084  
 dispose(Color) - of org.havi.ui.HGraphicsConfiguration 914  
 dispose(HScene) - of org.havi.ui.HSceneFactory 1034  
 DocumentAction - of org.dvb.dom.bootstrap 1307  
 DocumentFactory - of org.dvb.dom.bootstrap 1308  
 DOM - of org.dvb.dom.bootstrap.DocumentFactory 1308  
 DONT\_CARE - of org.havi.ui.HScreenConfigTemplate 1047  
 downloadFont(Font) - of org.havi.ui.HFontCapabilities 893  
 DripFeedDataSource - of org.dvb.media 584  
 DripFeedDataSource() - of org.dvb.media.DripFeedDataSource 584  
 DripFeedPermission - of org.dvb.media 587  
 DripFeedPermission(String) - of org.dvb.media.DripFeedPermission 587  
 DripFeedPermission(String, String) - of org.dvb.media.DripFeedPermission 587  
 DSMCCException - of org.dvb.dsmcc 630  
 DSMCCException() - of org.dvb.dsmcc.DSMCCException 630  
 DSMCCException(String) - of org.dvb.dsmcc.DSMCCException 630  
 DSMCCObject - of org.dvb.dsmcc 631  
 DSMCCObject(DSMCCObject, String) - of org.dvb.dsmcc.DSMCCObject 632  
 DSMCCObject(String) - of org.dvb.dsmcc.DSMCCObject 632  
 DSMCCObject(String, String) - of org.dvb.dsmcc.DSMCCObject 632  
 DSMCCStream - of org.dvb.dsmcc 637  
 DSMCCStream(DSMCCObject) - of org.dvb.dsmcc.DSMCCStream 637  
 DSMCCStream(String) - of org.dvb.dsmcc.DSMCCStream 637  
 DSMCCStream(String, String) - of org.dvb.dsmcc.DSMCCStream 638  
 DSMCCStreamEvent - of org.dvb.dsmcc 641  
 DSMCCStreamEvent(DSMCCObject) - of org.dvb.dsmcc.DSMCCStreamEvent 641  
 DSMCCStreamEvent(String) - of org.dvb.dsmcc.DSMCCStreamEvent 641  
 DSMCCStreamEvent(String, String) - of org.dvb.dsmcc.DSMCCStreamEvent 641  
 DST\_IN - of org.dvb.ui.DVBAlphaComposite 756  
 DST\_OUT - of org.dvb.ui.DVBAlphaComposite 756  
 DST\_OVER - of org.dvb.ui.DVBAlphaComposite 756  
 DstIn - of org.dvb.ui.DVBAlphaComposite 756  
 DstOut - of org.dvb.ui.DVBAlphaComposite 757  
 DstOver - of org.dvb.ui.DVBAlphaComposite 757  
 DVB\_HTML\_application - of org.dvb.application.AppAttributes 713  
 DVB\_J\_application - of org.dvb.application.AppAttributes 713  
 DVBAlphaComposite - of org.dvb.ui 755  
 DVBBufferedImage - of org.dvb.ui 760  
 DVBBufferedImage(int, int) - of org.dvb.ui.DVBBufferedImage 761  
 DVBBufferedImage(int, int, int) - of org.dvb.ui.DVBBufferedImage 761  
 DVBClassLoader - of org.dvb.lang 451  
 DVBClassLoader(URL[]) - of org.dvb.lang.DVBClassLoader 451  
 DVBClassLoader(URL[], ClassLoader) - of org.dvb.lang.DVBClassLoader 451  
 DVBColor - of org.dvb.ui 767  
 DVBColor(Color) - of org.dvb.ui.DVBColor 767  
 DVBColor(float, float, float, float) - of org.dvb.ui.DVBColor 767  
 DVBColor(int, boolean) - of org.dvb.ui.DVBColor 768  
 DVBColor(int, int, int, int) - of org.dvb.ui.DVBColor 768  
 DVBGraphics - of org.dvb.ui 771

DVBGraphics() - of org.dvb.ui.DVBGraphics 771  
DVBHTMLProxy - of org.dvb.application 740  
DVBJProxy - of org.dvb.application 742  
DvbNetworkInterfaceSIUtil - of org.dvb.net.tuning 751  
DVBRasterFormatException - of org.dvb.ui 775  
DVBRasterFormatException(String) - of org.dvb.ui.DVBRasterFormatException 775  
DVBScene - of org.dvb.application.inner 1316  
DVBTTest - of org.dvb.test 1255  
DVBTLayoutManager - of org.dvb.ui 776  
DVBTLayoutManager() - of org.dvb.ui.DVBTLayoutManager 778  
DVBTLayoutManager(int, int, int, int, boolean, int, int, int) - of org.dvb.ui.DVBTLayoutManager 778  
ECHO\_CHAR\_CHANGE - of org.havi.ui.HVisible 1180  
echoCharIsSet() - of org.havi.ui.HSinglelineEntry 1071  
EDIT\_MODE\_CHANGE - of org.havi.ui.HVisible 1180  
EMAIL\_CLIENT - of org.dvb.internet.InternetServiceFilter 1384  
EmailClient - of org.dvb.internet 1372  
EmailClientService - of org.dvb.internet 1373  
enableShortcuts(boolean) - of org.dvb.application.inner.DVBScene 1316  
enableShortcuts(boolean) - of org.havi.ui.HScene 1025  
ENCLOSED\_ALPHANUMERIC - of org.havi.ui.HFontCapabilities 888  
ENCLOSED\_CJK\_LETTERS\_AND\_MONTHS - of org.havi.ui.HFontCapabilities 888  
entryAdded(AppsDatabaseEvent) - of org.dvb.application.AppsDatabaseEventListener 733  
entryChanged(AppsDatabaseEvent) - of org.dvb.application.AppsDatabaseEventListener 733  
EntryExistsException - of org.dvb.internet 1374  
EntryExistsException() - of org.dvb.internet.EntryExistsException 1374  
EntryExistsException(String) - of org.dvb.internet.EntryExistsException 1374  
entryRemoved(AppsDatabaseEvent) - of org.dvb.application.AppsDatabaseEventListener 733  
equals(Object) - of org.dvb.application.AppsControlPermission 725  
equals(Object) - of org.dvb.ui.DVBAlphaComposite 758  
equals(Object) - of org.dvb.ui.DVBColor 769  
ER\_TYPE\_COLOR - of org.havi.ui.event.HEventRepresentation 1209  
ER\_TYPE\_NOT\_SUPPORTED - of org.havi.ui.event.HEventRepresentation 1209  
ER\_TYPE\_STRING - of org.havi.ui.event.HEventRepresentation 1209  
ER\_TYPE\_SYMBOL - of org.havi.ui.event.HEventRepresentation 1210  
EventManager - of org.dvb.event 457  
EXTENDED\_EVENT - of org.dvb.si.DescriptorTag 497  
ExtendedAppAttributes - of org.dvb.application.storage 1355  
Facility - of org.dvb.user 479  
Facility(String, String) - of org.dvb.user.Facility 479  
Facility(String, String[]) - of org.dvb.user.Facility 479  
FAIL - of org.dvb.test.DVBTTest 1255  
feed(byte[]) - of org.dvb.media.DripFeedDataSource 585  
FileAccessPermissions - of org.dvb.io.persistent 471  
FileAccessPermissions(boolean, boolean, boolean, boolean, boolean, boolean) - of org.dvb.io.persistent.FileAccessPermissions 471  
FileAttributes - of org.dvb.io.persistent 473  
findClass(String) - of org.dvb.lang.DVBClassLoader 452  
FIRST\_CHANGE - of org.havi.ui.HVisible 1180  
FIRST\_STATE - of org.havi.ui.HState 1089  
FLICKER\_FILTERING - of org.havi.ui.HScreenConfigTemplate 1047  
flush() - of org.dvb.ui.DVBBufferedImage 762  
flush() - of org.havi.ui.HBackgroundImage 840  
FM\_RADIO - of org.dvb.si.SIServiceType 561  
FOCUS\_TRANSFER - of org.havi.ui.event.HFocusEvent 1213  
FOCUSED\_STATE - of org.havi.ui.HState 1090

FOCUSED\_STATE\_BIT - of org.havi.ui.HState 1090  
 focusGained(FocusEvent) - of org.havi.ui.HEventMulticaster 870  
 focusLost(FocusEvent) - of org.havi.ui.HEventMulticaster 870  
 fontAccessible(Font) - of org.havi.ui.HFontCapabilities 893  
 FontFactory - of org.dvb.ui 783  
 FontFactory() - of org.dvb.ui.FontFactory 783  
 FontFactory(URL) - of org.dvb.ui.FontFactory 783  
 FontFormatException - of org.dvb.ui 785  
 FontFormatException() - of org.dvb.ui.FontFormatException 785  
 FontFormatException(String) - of org.dvb.ui.FontFormatException 785  
 FontNotAvailableException - of org.dvb.ui 786  
 FontNotAvailableException() - of org.dvb.ui.FontNotAvailableException 786  
 FontNotAvailableException(String) - of org.dvb.ui.FontNotAvailableException 786  
 FREQUENCY\_LIST - of org.dvb.si.DescriptorTag 497  
 FROM\_CACHE - of org.dvb.dsmcc.DSMCCObject 631  
 FROM\_CACHE\_ONLY - of org.dvb.si.SIInformation 531  
 FROM\_CACHE\_OR\_STREAM - of org.dvb.dsmcc.DSMCCObject 631  
 FROM\_CACHE\_OR\_STREAM - of org.dvb.si.SIInformation 531  
 FROM\_STREAM\_ONLY - of org.dvb.dsmcc.DSMCCObject 632  
 FROM\_STREAM\_ONLY - of org.dvb.si.SIInformation 531  
 fromActual() - of org.dvb.si.SIInformation 532  
 GENERAL\_PUNCTUATION - of org.havi.ui.HFontCapabilities 888  
 GeneralPreference - of org.dvb.user 480  
 GeneralPreference(String) - of org.dvb.user.GeneralPreference 480  
 GEOMETRICAL\_SHAPES - of org.havi.ui.HFontCapabilities 888  
 GEORGIAN\_EXTENDED - of org.havi.ui.HFontCapabilities 888  
 getAcceptedMediaTypes() - of org.dvb.internet.WWWBrowserService 1394  
 getActionCommand() - of org.havi.ui.event.HActionEvent 1196  
 getActionCommand() - of org.havi.ui.HActionable 802  
 getActionCommand() - of org.havi.ui.HGraphicButton 898  
 getActionCommand() - of org.havi.ui.HTextButton 1126  
 getActionCommand() - of org.havi.ui.HToggleButton 1146  
 getActions() - of org.dvb.application.AppsControlPermission 726  
 getActionSound() - of org.havi.ui.HActionable 802  
 getActionSound() - of org.havi.ui.HGraphicButton 898  
 getActionSound() - of org.havi.ui.HTextButton 1126  
 getActionSound() - of org.havi.ui.HToggleButton 1146  
 getActiveFormatDefinition() - of org.dvb.media.VideoFormatControl 605  
 getActiveVideoArea() - of org.dvb.media.VideoPresentationControl 610  
 getActiveVideoAreaOnScreen() - of org.dvb.media.VideoPresentationControl 610  
 getAdjustmentSound() - of org.havi.ui.HAdjustmentValue 812  
 getAdjustmentSound() - of org.havi.ui.HRangeValue 1014  
 getAdjustMode() - of org.havi.ui.HAdjustmentInputPreferred 808  
 getAdjustMode() - of org.havi.ui.HRangeValue 1014  
 getAID() - of org.dvb.application.AppID 719  
 getAllFonts() - of org.havi.ui.HGraphicsConfiguration 914  
 getAllShortcutKeycodes() - of org.dvb.application.inner.DVBScene 1316  
 getAllShortcutKeycodes() - of org.havi.ui.HScene 1025  
 getAlpha() - of org.dvb.ui.DVBAlphaComposite 759  
 getAlpha() - of org.dvb.ui.DVBColor 769  
 getAnimateContent(int) - of org.havi.ui.HVisible 1185  
 getAppAttributes(AppID) - of org.dvb.application.AppsDatabase 728  
 getAppAttributes(AppsDatabaseFilter) - of org.dvb.application.AppsDatabase 728  
 getAppData() - of org.dvb.si.SIRetrievalEvent 551  
 getAppIcon() - of org.dvb.application.AppAttributes 713

getAppID() - of org.dvb.application.AppsDatabaseEvent 732  
getAppID() - of org.dvb.application.AppStateChangeEvent 737  
getAppIDs(AppsDatabaseFilter) - of org.dvb.application.AppsDatabase 729  
getAppProxy(AppID) - of org.dvb.application.AppsDatabase 729  
getAppsDatabase() - of org.dvb.application.AppsDatabase 729  
getAspectRatio() - of org.dvb.media.VideoFormatControl 605  
getAvailableCompositeRules() - of org.dvb.ui.DVBGraphics 772  
getBackgroundImage() - of org.dvb.application.inner.DVBScene 1316  
getBackgroundImage() - of org.havi.ui.HScene 1025  
getBackgroundMode() - of org.dvb.application.inner.DVBScene 1317  
getBackgroundMode() - of org.havi.ui.HScene 1026  
getBackgroundMode() - of org.havi.ui.HVisible 1185  
getBehavior() - of org.havi.ui.HStaticRange 1102  
getBestColorMatch(Color) - of org.dvb.ui.DVBGraphics 772  
getBestConfiguration(HBackgroundConfigTemplate) - of org.havi.ui.HBackgroundDevice 836  
getBestConfiguration(HBackgroundConfigTemplate[]) - of org.havi.ui.HBackgroundDevice 836  
getBestConfiguration(HBackgroundConfigTemplate[]) - of org.havi.ui.HScreen 1041  
getBestConfiguration(HGraphicsConfigTemplate) - of org.havi.ui.HGraphicsDevice 919  
getBestConfiguration(HGraphicsConfigTemplate[]) - of org.havi.ui.HGraphicsDevice 919  
getBestConfiguration(HGraphicsConfigTemplate[]) - of org.havi.ui.HScreen 1042  
getBestConfiguration(HVideoConfigTemplate) - of org.havi.ui.HVideoDevice 1169  
getBestConfiguration(HVideoConfigTemplate[]) - of org.havi.ui.HScreen 1042  
getBestConfiguration(HVideoConfigTemplate[]) - of org.havi.ui.HVideoDevice 1169  
getBestScene(HSceneTemplate) - of org.havi.ui.HSceneFactory 1034  
getBestSceneTemplate(HSceneTemplate) - of org.havi.ui.HSceneFactory 1035  
getBlockIncrement() - of org.havi.ui.HAdjustmentValue 812  
getBlockIncrement() - of org.havi.ui.HRangeValue 1014  
getBouquetID() - of org.dvb.si.SIBouquet 505  
getBouquetID() - of org.dvb.si.SIMonitoringEvent 539  
getBouquetID() - of org.dvb.si.SITransportStreamBAT 569  
getByteAt(int) - of org.dvb.si.Descriptor 494  
getCaretCharPosition() - of org.havi.ui.HSinglelineEntry 1071  
getCaretCharPositionForLine(HVisible, int) - of org.havi.ui.HMultilineEntryLook 982  
getCaretPositionNextLine(HVisible) - of org.havi.ui.HMultilineEntryLook 982  
getCaretPositionPreviousLine(HVisible) - of org.havi.ui.HMultilineEntryLook 982  
getCarouselId() - of org.dvb.dsmcc.ServiceXFRReference 678  
getClassLoader(String[]) - of org.dvb.application.plugins.ApplicationSecurityContext 1331  
getClient() - of org.dvb.event.RepositoryDescriptor 461  
getClient() - of org.dvb.net.rc.ConnectionRCInterface 697  
getClient() - of org.havi.ui.HScreenDevice 1055  
getClipRegion() - of org.dvb.media.VideoPresentationControl 610  
getClipRegion() - of org.dvb.media.VideoTransformation 614  
getClosestMatch(VideoTransformation) - of org.dvb.media.BackgroundVideoPresentationControl 579  
getCode() - of org.dvb.event.UserEvent 463  
getCoherentScreenConfigurations(HScreenConfigTemplate[]) - of org.havi.ui.HScreen 1043  
getColor() - of org.dvb.ui.DVBGraphics 772  
getColor() - of org.havi.ui.event.HEventRepresentation 1210  
getColor() - of org.havi.ui.HBackgroundConfiguration 834  
getCompatibleImage(Image, HImageHints) - of org.havi.ui.HGraphicsConfiguration 914  
getComponentHScreenRectangle(Component) - of org.havi.ui.HGraphicsConfiguration 915  
getComponentTag() - of org.dvb.si.PMTElementaryStream 500  
getConfigTemplate() - of org.havi.ui.HBackgroundConfiguration 834  
getConfigTemplate() - of org.havi.ui.HEmulatedGraphicsConfiguration 862  
getConfigTemplate() - of org.havi.ui.HGraphicsConfiguration 915  
getConfigTemplate() - of org.havi.ui.HVideoConfiguration 1167

getConfigurations() - of org.havi.ui.HBackgroundDevice 837  
getConfigurations() - of org.havi.ui.HGraphicsDevice 920  
getConfigurations() - of org.havi.ui.HVideoDevice 1170  
getConnectedTime() - of org.dvb.net.rc.ConnectionRCInterface 697  
getContent() - of org.dvb.si.Descriptor 494  
getContent(int) - of org.havi.ui.HVisible 1185  
getContentLength() - of org.dvb.si.Descriptor 494  
getContentNibbles() - of org.dvb.si.SIEvent 525  
getContentType() - of org.dvb.media.DripFeedDataSource 585  
getControl(String) - of org.dvb.media.DripFeedDataSource 585  
getControls() - of org.dvb.media.DripFeedDataSource 586  
getCurrent() - of org.havi.ui.HToggleGroup 1154  
getCurrentConfiguration() - of org.havi.ui.HBackgroundDevice 837  
getCurrentConfiguration() - of org.havi.ui.HGraphicsDevice 920  
getCurrentConfiguration() - of org.havi.ui.HVideoDevice 1170  
getCurrentIndex() - of org.havi.ui.HListGroup 950  
getCurrentItem() - of org.havi.ui.HListGroup 951  
getCurrentTarget() - of org.dvb.net.rc.ConnectionRCInterface 697  
getCurrentVersionNumber() - of org.dvb.application.storage.ExtendedAppAttributes 1355  
getDataRate() - of org.dvb.net.rc.RCInterface 704  
getDataSource() - of org.dvb.si.SIInformation 532  
getDecoderFormatConversion() - of org.dvb.media.VideoFormatControl 606  
getDefaultConfiguration() - of org.havi.ui.HBackgroundDevice 837  
getDefaultConfiguration() - of org.havi.ui.HGraphicsDevice 920  
getDefaultConfiguration() - of org.havi.ui.HVideoDevice 1170  
getDefaultHBackgroundDevice() - of org.havi.ui.HScreen 1043  
getDefaultHGraphicsDevice() - of org.havi.ui.HScreen 1044  
getDefaultHScene() - of org.havi.ui.HSceneFactory 1035  
getDefaultHScene(HScreen) - of org.havi.ui.HSceneFactory 1035  
getDefaultHScreen() - of org.havi.ui.HScreen 1044  
getDefaultHVideoDevice() - of org.havi.ui.HScreen 1044  
getDefaultLook() - of org.havi.ui.HAnimation 826  
getDefaultLook() - of org.havi.ui.HGraphicButton 898  
getDefaultLook() - of org.havi.ui.HIcon 924  
getDefaultLook() - of org.havi.ui.HListGroup 951  
getDefaultLook() - of org.havi.ui.HMultilineEntry 980  
getDefaultLook() - of org.havi.ui.HRange 1001  
getDefaultLook() - of org.havi.ui.HRangeValue 1014  
getDefaultLook() - of org.havi.ui.HSinglelineEntry 1071  
getDefaultLook() - of org.havi.ui.HStaticAnimation 1093  
getDefaultLook() - of org.havi.ui.HStaticIcon 1099  
getDefaultLook() - of org.havi.ui.HStaticRange 1103  
getDefaultLook() - of org.havi.ui.HStaticText 1109  
getDefaultLook() - of org.havi.ui.HText 1119  
getDefaultLook() - of org.havi.ui.HTextButton 1127  
getDefaultLook() - of org.havi.ui.HToggleButton 1146  
getDefaultSize() - of org.havi.ui.HVisible 1186  
getDelay() - of org.havi.ui.HAnimateEffect 815  
getDelay() - of org.havi.ui.HFlatEffectMatte 878  
getDelay() - of org.havi.ui.HImageEffectMatte 929  
getDelay() - of org.havi.ui.HStaticAnimation 1093  
getDenominator() - of org.dvb.dsmcc.NPTRate 662  
getDescriptorTags() - of org.dvb.si.SIBouquet 505  
getDescriptorTags() - of org.dvb.si.SIInformation 532  
getDescriptorTags() - of org.dvb.si.SINetwork 543

getDevice() - of org.havi.ui.HBackgroundConfiguration 834  
getDevice() - of org.havi.ui.HGraphicsConfiguration 915  
getDevice() - of org.havi.ui.HVideoConfiguration 1167  
getDisplayAspectRatio() - of org.dvb.media.VideoFormatControl 606  
getDNSServer() - of org.dvb.net.rc.ConnectionParameters 693  
getDuration() - of org.dvb.dsmcc.DSMCCStream 638  
getDuration() - of org.dvb.media.DripFeedDataSource 586  
getDuration() - of org.dvb.si.SIEvent 525  
getDVBComposite() - of org.dvb.ui.DVBGraphics 773  
getDvbLocator() - of org.dvb.si.PMTElementaryStream 500  
getDvbLocator() - of org.dvb.si.PMTService 502  
getDvbLocator() - of org.dvb.si.SIEvent 525  
getDvbLocator() - of org.dvb.si.SIService 555  
getDvbLocator() - of org.dvb.si.SITransportStream 567  
getEchoChar() - of org.havi.ui.HSinglelineEntry 1071  
getEditMode() - of org.havi.ui.HKeyboardInputPreferred 942  
getEditMode() - of org.havi.ui.HSinglelineEntry 1071  
getEITPresentFollowingFlag() - of org.dvb.si.SIService 555  
getEITScheduleFlag() - of org.dvb.si.SIService 555  
getElementaryPID() - of org.dvb.si.PMTElementaryStream 500  
getElementaryStreams() - of org.davic.mpeg.NotAuthorizedException 374  
getElementInsets() - of org.havi.ui.HListGroupLook 962  
getEmulation() - of org.havi.ui.HEmulatedGraphicsConfiguration 862  
getEndTime() - of org.dvb.si.SIMonitoringEvent 539  
getEventData() - of org.dvb.dsmcc.StreamEvent 680  
getEventId() - of org.dvb.application.AppsDatabaseEvent 732  
getEventId() - of org.dvb.dsmcc.StreamEvent 681  
getEventID() - of org.dvb.si.SIEvent 526  
getEventList() - of org.dvb.dsmcc.DSMCCStreamEvent 642  
getEventName() - of org.dvb.dsmcc.StreamEvent 681  
getEventNPT() - of org.dvb.dsmcc.StreamEvent 681  
getExpirationDate() - of org.dvb.io.persistent.FileAttributes 473  
getFamily() - of org.dvb.event.UserEvent 463  
getFavourites() - of org.dvb.user.Preference 482  
getFileAttributes(File) - of org.dvb.io.persistent.FileAttributes 473  
getFirstNPT() - of org.dvb.dsmcc.NPTDiscontinuityEvent 659  
getFlickerFilter() - of org.havi.ui.HScreenConfiguration 1052  
getFocusOwner() - of org.dvb.application.inner.DVBScene 1317  
getFocusOwner() - of org.havi.ui.HScene 1026  
getForcedSelection() - of org.havi.ui.HToggleGroup 1154  
getFreeCAMode() - of org.dvb.si.SIEvent 526  
getFreeCAMode() - of org.dvb.si.SIService 555  
getFromState() - of org.dvb.application.AppStateChangeEvent 737  
getFullScreenScene(HGraphicsDevice) - of org.havi.ui.HSceneFactory 1036  
getGainFocusSound() - of org.dvb.application.inner.InnerApplicationContainer 1322  
getGainFocusSound() - of org.havi.ui.HAnimation 826  
getGainFocusSound() - of org.havi.ui.HGraphicButton 898  
getGainFocusSound() - of org.havi.ui.HIcon 924  
getGainFocusSound() - of org.havi.ui.HListGroup 951  
getGainFocusSound() - of org.havi.ui.HNavigable 989  
getGainFocusSound() - of org.havi.ui.HRange 1002  
getGainFocusSound() - of org.havi.ui.HRangeValue 1014  
getGainFocusSound() - of org.havi.ui.HSinglelineEntry 1072  
getGainFocusSound() - of org.havi.ui.HText 1119  
getGainFocusSound() - of org.havi.ui.HTextButton 1127

getGainFocusSound() - of org.havi.ui.HToggleButton 1147  
getGraphicContent(int) - of org.havi.ui.HVisible 1186  
getGraphics() - of org.dvb.ui.DVBBufferedImage 762  
getHBackgroundDevices() - of org.havi.ui.HScreen 1044  
getHeight() - of org.dvb.ui.DVBBufferedImage 762  
getHeight() - of org.havi.ui.HBackgroundImage 840  
getHeight(ImageObserver) - of org.dvb.ui.DVBBufferedImage 762  
getHGraphicsDevices() - of org.havi.ui.HScreen 1044  
getHorizontalAlign() - of org.dvb.ui.DVBTextLayoutManager 778  
getHorizontalAlignment() - of org.havi.ui.HVisible 1186  
getHorizontalScalingFactors() - of org.dvb.media.VideoPresentationControl 611  
getHorizontalTabSpacing() - of org.dvb.ui.DVBTextLayoutManager 778  
getHScreens() - of org.havi.ui.HScreen 1044  
getHVideoDevices() - of org.havi.ui.HScreen 1045  
getIcon() - of org.havi.ui.HListElement 945  
getIconFlags() - of org.dvb.application.AppIcon 718  
getIconSize() - of org.havi.ui.HListGroup 951  
getID() - of org.havi.ui.event.HBackgroundImageEvent 1204  
getIdentifier() - of org.dvb.application.AppAttributes 713  
getIDstring() - of org.havi.ui.HScreenDevice 1056  
getImage() - of org.dvb.ui.DVBBufferedImage 762  
getImplementation() - of org.havi.ui.HEmulatedGraphicsConfiguration 862  
getIndex(HListElement) - of org.havi.ui.HListGroup 951  
getInputDeviceSupported() - of org.havi.ui.event.HKeyCapabilities 1221  
getInputDeviceSupported() - of org.havi.ui.event.HMouseCapabilities 1226  
getInputDeviceSupported() - of org.havi.ui.event.HRcCapabilities 1227  
getInputVideoSize() - of org.dvb.media.VideoPresentationControl 611  
getInsets() - of org.dvb.ui.DVBTextLayoutManager 779  
getInsets(HVisible) - of org.havi.ui.HAnimateLook 819  
getInsets(HVisible) - of org.havi.ui.HGraphicLook 905  
getInsets(HVisible) - of org.havi.ui.HListGroupLook 962  
getInsets(HVisible) - of org.havi.ui.HLook 969  
getInsets(HVisible) - of org.havi.ui.HMultilineEntryLook 983  
getInsets(HVisible) - of org.havi.ui.HRangeLook 1007  
getInsets(HVisible) - of org.havi.ui.HSinglelineEntryLook 1080  
getInsets(HVisible) - of org.havi.ui.HTextLook 1136  
getInstance() - of org.dvb.event.EventManager 458  
getInstance() - of org.dvb.net.rc.RCInterfaceManager 705  
getInstance() - of org.dvb.user.UserPreferenceManager 487  
getInstance() - of org.havi.ui.HSceneFactory 1036  
getInstance(int) - of org.dvb.ui.DVBAlphaComposite 759  
getInstance(int, float) - of org.dvb.ui.DVBAlphaComposite 759  
getInteractionState() - of org.havi.ui.HVisible 1186  
getInterface(InetAddress) - of org.dvb.net.rc.RCInterfaceManager 705  
getInterface(Socket) - of org.dvb.net.rc.RCInterfaceManager 706  
getInterface(URLConnection) - of org.dvb.net.rc.RCInterfaceManager 706  
getInterfaces() - of org.dvb.net.rc.RCInterfaceManager 706  
getInterlaced() - of org.havi.ui.HScreenConfiguration 1053  
getInvocationHandler(Object) - of org.dvb.application.plugins.Proxy 1338  
getIsServiceBound() - of org.dvb.application.AppAttributes 714  
getItem() - of org.havi.ui.event.HItemEvent 1219  
getItem(int) - of org.havi.ui.HListGroup 952  
getKeyChar() - of org.dvb.event.UserEvent 463  
getLabel() - of org.havi.ui.HListElement 945  
getLabelSize() - of org.havi.ui.HListGroup 952

getLastNPT() - of org.dvb.dsmcc.NPTDiscontinuityEvent 659  
getLetterSpace() - of org.dvb.ui.DVBTextLayoutManager 779  
getLevel1ContentNibbles() - of org.dvb.si.SIEvent 526  
getLineOrientation() - of org.dvb.ui.DVBTextLayoutManager 779  
getLineSpace() - of org.dvb.ui.DVBTextLayoutManager 779  
getListContent() - of org.havi.ui.HListGroup 952  
getLocator() - of org.dvb.application.AppIcon 718  
getLocator() - of org.dvb.application.storage.StoredApplicationService 1360  
getLocator() - of org.dvb.dsmcc.ServiceDomain 673  
getLocator() - of org.dvb.dsmcc.ServiceXFRReference 679  
getLook() - of org.havi.ui.HVisible 1187  
getLookData(Object) - of org.havi.ui.HVisible 1187  
getLoseFocusSound() - of org.dvb.application.inner.InnerApplicationContainer 1322  
getLoseFocusSound() - of org.havi.ui.HAnimation 826  
getLoseFocusSound() - of org.havi.ui.HGraphicButton 899  
getLoseFocusSound() - of org.havi.ui.HIcon 925  
getLoseFocusSound() - of org.havi.ui.HListGroup 952  
getLoseFocusSound() - of org.havi.ui.HNavigable 989  
getLoseFocusSound() - of org.havi.ui.HRange 1002  
getLoseFocusSound() - of org.havi.ui.HRangeValue 1015  
getLoseFocusSound() - of org.havi.ui.HSinglelineEntry 1072  
getLoseFocusSound() - of org.havi.ui.HText 1120  
getLoseFocusSound() - of org.havi.ui.HTextButton 1127  
getLoseFocusSound() - of org.havi.ui.HToggleButton 1147  
getMatte() - of org.havi.ui.HComponent 844  
getMatte() - of org.havi.ui.HContainer 854  
getMatte() - of org.havi.ui.HMatteLayer 976  
getMatteData() - of org.havi.ui.HFlatEffectMatte 878  
getMatteData() - of org.havi.ui.HFlatMatte 883  
getMatteData() - of org.havi.ui.HImageEffectMatte 929  
getMatteData() - of org.havi.ui.HImageMatte 936  
getMaxCharsPerLine() - of org.havi.ui.HSinglelineEntry 1072  
getMaximumSize() - of org.havi.ui.HVisible 1187  
getMaximumSize(HVisible) - of org.havi.ui.HAnimateLook 819  
getMaximumSize(HVisible) - of org.havi.ui.HGraphicLook 905  
getMaximumSize(HVisible) - of org.havi.ui.HListGroupLook 963  
getMaximumSize(HVisible) - of org.havi.ui.HLook 969  
getMaximumSize(HVisible) - of org.havi.ui.HMultilineEntryLook 983  
getMaximumSize(HVisible) - of org.havi.ui.HRangeLook 1007  
getMaximumSize(HVisible) - of org.havi.ui.HSinglelineEntryLook 1080  
getMaximumSize(HVisible) - of org.havi.ui.HTextLook 1136  
getMaxValue() - of org.havi.ui.HStaticRange 1103  
getMinimumSize() - of org.havi.ui.HVisible 1187  
getMinimumSize(HVisible) - of org.havi.ui.HAnimateLook 820  
getMinimumSize(HVisible) - of org.havi.ui.HGraphicLook 906  
getMinimumSize(HVisible) - of org.havi.ui.HListGroupLook 963  
getMinimumSize(HVisible) - of org.havi.ui.HLook 970  
getMinimumSize(HVisible) - of org.havi.ui.HMultilineEntryLook 983  
getMinimumSize(HVisible) - of org.havi.ui.HRangeLook 1007  
getMinimumSize(HVisible) - of org.havi.ui.HSinglelineEntryLook 1080  
getMinimumSize(HVisible) - of org.havi.ui.HTextLook 1137  
getMinValue() - of org.havi.ui.HStaticRange 1103  
getModifiers() - of org.dvb.event.UserEvent 463  
getModifiers() - of org.havi.ui.event.HActionEvent 1197  
getMostFavourite() - of org.dvb.user.Preference 482



getMountPoint() - of org.dvb.dsmcc.ServiceDomain 673  
getMove(int) - of org.dvb.application.inner.InnerApplicationContainer 1322  
getMove(int) - of org.havi.ui.HAnimation 826  
getMove(int) - of org.havi.ui.HGraphicButton 899  
getMove(int) - of org.havi.ui.HIcon 925  
getMove(int) - of org.havi.ui.HListGroup 952  
getMove(int) - of org.havi.ui.HNavigable 989  
getMove(int) - of org.havi.ui.HRange 1002  
getMove(int) - of org.havi.ui.HRangeValue 1015  
getMove(int) - of org.havi.ui.HSinglelineEntry 1072  
getMove(int) - of org.havi.ui.HText 1120  
getMove(int) - of org.havi.ui.HTextButton 1127  
getMove(int) - of org.havi.ui.HToggleButton 1147  
getMultiSelection() - of org.havi.ui.HListGroup 953  
getName() - of org.dvb.application.AppAttributes 714  
getName() - of org.dvb.application.storage.StoredApplicationService 1361  
getName() - of org.dvb.event.RepositoryDescriptor 461  
getName() - of org.dvb.internet.InternetClientService 1381  
getName() - of org.dvb.si.SIBouquet 505  
getName() - of org.dvb.si.SIEvent 526  
getName() - of org.dvb.si.SINetwork 543  
getName() - of org.dvb.si.SIService 556  
getName() - of org.dvb.user.Preference 482  
getName() - of org.dvb.user.UserPreferenceChangeEvent 485  
getName(String) - of org.dvb.application.AppAttributes 714  
getNames() - of org.dvb.application.AppAttributes 714  
getNavigationKeys() - of org.dvb.application.inner.InnerApplicationContainer 1322  
getNavigationKeys() - of org.havi.ui.HAnimation 827  
getNavigationKeys() - of org.havi.ui.HGraphicButton 899  
getNavigationKeys() - of org.havi.ui.HIcon 925  
getNavigationKeys() - of org.havi.ui.HListGroup 953  
getNavigationKeys() - of org.havi.ui.HNavigationInputPreferred 992  
getNavigationKeys() - of org.havi.ui.HRange 1002  
getNavigationKeys() - of org.havi.ui.HRangeValue 1015  
getNavigationKeys() - of org.havi.ui.HSinglelineEntry 1073  
getNavigationKeys() - of org.havi.ui.HText 1120  
getNavigationKeys() - of org.havi.ui.HTextButton 1128  
getNavigationKeys() - of org.havi.ui.HToggleButton 1147  
getNetworkID() - of org.dvb.si.SIMonitoringEvent 539  
getNetworkID() - of org.dvb.si.SINetwork 544  
getNetworkID() - of org.dvb.si.SITransportStreamNIT 571  
getNetworkInterface(SIDatabase) - of org.dvb.net.tuning.DvbNetworkInterfaceSIUtil 751  
getNewDFC() - of org.dvb.media.DFCChangedEvent 583  
getNewFormat() - of org.dvb.media.ActiveFormatDescriptionChangedEvent 577  
getNewRatio() - of org.dvb.media.AspectRatioChangedEvent 578  
getNewVersionNumber() - of org.dvb.dsmcc.ObjectChangeEvent 666  
getNPT() - of org.dvb.dsmcc.DSMCCStream 638  
getNPTRate() - of org.dvb.dsmcc.DSMCCStream 639  
getNSAPAddress() - of org.dvb.dsmcc.ServiceDomain 673  
getNSAPAddress() - of org.dvb.dsmcc.ServiceXFRReference 679  
getNumerator() - of org.dvb.dsmcc.NPTRate 662  
getNumItems() - of org.havi.ui.HListGroup 953  
getNumSelected() - of org.havi.ui.HListGroup 953  
getNumVisible(HVisible) - of org.havi.ui.HListGroupLook 964  
getOffset() - of org.havi.ui.HImageMatte 936

getOffset(HScreenConfiguration) - of org.havi.ui.HScreenConfiguration 1053  
 getOffset(int) - of org.havi.ui.HImageEffectMatte 929  
 getOID() - of org.dvb.application.AppID 719  
 getOrganisationId() - of org.dvb.application.storage.StoredApplicationService 1361  
 getOrientation() - of org.havi.ui.HListGroup 953  
 getOrientation() - of org.havi.ui.HOrientable 996  
 getOrientation() - of org.havi.ui.HStaticRange 1103  
 getOriginalNetworkID() - of org.dvb.si.PMTElementaryStream 501  
 getOriginalNetworkID() - of org.dvb.si.PMTService 502  
 getOriginalNetworkID() - of org.dvb.si.SIEvent 526  
 getOriginalNetworkID() - of org.dvb.si.SIMonitoringEvent 539  
 getOriginalNetworkID() - of org.dvb.si.SIService 556  
 getOriginalNetworkID() - of org.dvb.si.SITransportStream 567  
 getPassword() - of org.dvb.net.rc.ConnectionParameters 694  
 getPathName() - of org.dvb.dsmcc.ServiceXFRReference 679  
 getPcrPid() - of org.dvb.si.PMTService 502  
 getPermissions() - of org.dvb.io.persistent.FileAttributes 474  
 getPixelAspectRatio() - of org.havi.ui.HScreenConfiguration 1053  
 getPixelCoordinatesHScreenRectangle(HScreenRectangle, Container) - of org.havi.ui.HGraphicsConfigura-  
 tion 915  
 getPixelCoordinatesHScreenRectangle(Rectangle) - of org.havi.ui.HScene 1026  
 getPixelResolution() - of org.havi.ui.HScreenConfiguration 1053  
 getPlayMode() - of org.havi.ui.HAnimateEffect 815  
 getPlayMode() - of org.havi.ui.HFlatEffectMatte 878  
 getPlayMode() - of org.havi.ui.HImageEffectMatte 930  
 getPlayMode() - of org.havi.ui.HStaticAnimation 1094  
 getPosition() - of org.havi.ui.HAnimateEffect 815  
 getPosition() - of org.havi.ui.HFlatEffectMatte 879  
 getPosition() - of org.havi.ui.HImageEffectMatte 930  
 getPosition() - of org.havi.ui.HStaticAnimation 1094  
 getPosition(String) - of org.dvb.user.Preference 483  
 getPositioningCapability() - of org.dvb.media.VideoPresentationControl 611  
 getPreferenceObject(int) - of org.havi.ui.HGraphicsConfigTemplate 910  
 getPreferenceObject(int) - of org.havi.ui.HSceneTemplate 1039  
 getPreferenceObject(int) - of org.havi.ui.HScreenConfigTemplate 1050  
 getPreferenceObject(int) - of org.havi.ui.HVideoConfigTemplate 1164  
 getPreferencePriority(int) - of org.havi.ui.HBackgroundConfigTemplate 831  
 getPreferencePriority(int) - of org.havi.ui.HGraphicsConfigTemplate 911  
 getPreferencePriority(int) - of org.havi.ui.HSceneTemplate 1040  
 getPreferencePriority(int) - of org.havi.ui.HScreenConfigTemplate 1050  
 getPreferencePriority(int) - of org.havi.ui.HVideoConfigTemplate 1164  
 getPreferredSize() - of org.havi.ui.HVisible 1188  
 getPreferredSize(HVisible) - of org.havi.ui.HAnimateLook 820  
 getPreferredSize(HVisible) - of org.havi.ui.HGraphicLook 906  
 getPreferredSize(HVisible) - of org.havi.ui.HListGroupLook 964  
 getPreferredSize(HVisible) - of org.havi.ui.HLook 970  
 getPreferredSize(HVisible) - of org.havi.ui.HMultilineEntryLook 984  
 getPreferredSize(HVisible) - of org.havi.ui.HRangeLook 1008  
 getPreferredSize(HVisible) - of org.havi.ui.HSinglelineEntryLook 1081  
 getPreferredSize(HVisible) - of org.havi.ui.HTextLook 1137  
 getPriority() - of org.dvb.application.AppAttributes 715  
 getPriority() - of org.dvb.io.persistent.FileAttributes 474  
 getProfiles() - of org.dvb.application.AppAttributes 715  
 getProperty(String) - of org.dvb.application.AppAttributes 715  
 getProperty(String, ImageObserver) - of org.dvb.ui.DVBBufferedImage 763

getProviderName() - of org.dvb.si.SIService 556  
getProxyClass(ClassLoader, Class[]) - of org.dvb.application.plugins.Proxy 1338  
getPunchThroughToBackgroundColor(Color, int) - of org.havi.ui.HGraphicsConfiguration 916  
getPunchThroughToBackgroundColor(Color, int, HVideoDevice) - of org.havi.ui.HGraphicsConfiguration 916  
getPunchThroughToBackgroundColor(int) - of org.havi.ui.HGraphicsConfiguration 917  
getPunchThroughToBackgroundColor(int, HVideoDevice) - of org.havi.ui.HGraphicsConfiguration 917  
getRate() - of org.dvb.dsmcc.NPTRateChangeEvent 663  
getReason() - of org.dvb.media.PresentationChangedEvent 591  
getReason(int) - of org.davic.mpeg.NotAuthorizedException 375  
getReceiveBufferSize(DatagramSocket) - of org.dvb.net.DatagramSocketBufferControl 687  
getRenderMode() - of org.dvb.application.inner.DVBScene 1317  
getRenderMode() - of org.havi.ui.HScene 1026  
getRepeatCount() - of org.havi.ui.HAnimateEffect 815  
getRepeatCount() - of org.havi.ui.HFlatEffectMatte 879  
getRepeatCount() - of org.havi.ui.HImageEffectMatte 930  
getRepeatCount() - of org.havi.ui.HStaticAnimation 1094  
getRepresentation(int) - of org.havi.ui.event.HRcCapabilities 1228  
getResizeMode() - of org.havi.ui.HVisible 1188  
getResource(String, boolean) - of org.dvb.application.plugins.ApplicationSecurityContext 1331  
getResult() - of org.dvb.si.SISuccessfulRetrieveEvent 563  
getRGB() - of org.dvb.ui.DVBColor 769  
getRGB(int, int) - of org.dvb.ui.DVBBufferedImage 763  
getRGB(int, int, int, int, int[], int, int) - of org.dvb.ui.DVBBufferedImage 764  
getRootContainer(XletContext) - of org.dvb.application.plugins.XletSystemCall 1345  
getRule() - of org.dvb.ui.DVBAlphaComposite 759  
getRunningStatus() - of org.dvb.si.SIEvent 526  
getRunningStatus() - of org.dvb.si.SIService 556  
getScalingFactors() - of org.dvb.media.VideoTransformation 615  
getSceneTemplate() - of org.havi.ui.HScene 1026  
getScreenArea() - of org.havi.ui.HScreenConfiguration 1053  
getScreenAspectRatio() - of org.havi.ui.HScreenDevice 1056  
getScrollPosition() - of org.havi.ui.HListGroup 953  
getSelection() - of org.havi.ui.HListGroup 954  
getSelectionIndices() - of org.havi.ui.HListGroup 954  
getSelectionMode() - of org.havi.ui.HListGroup 954  
getSelectionMode() - of org.havi.ui.HSelectionInputPreferred 1065  
getSelectionSound() - of org.havi.ui.HItemValue 940  
getSelectionSound() - of org.havi.ui.HListGroup 954  
getService() - of org.davic.mpeg.NotAuthorizedException 375  
getService() - of org.dvb.internet.InternetClient 1376  
getServiceContentLocators() - of org.dvb.internet.InternetClient 1376  
getServiceId() - of org.dvb.application.storage.StoredApplicationService 1361  
getServiceID() - of org.dvb.si.PMTElementaryStream 501  
getServiceID() - of org.dvb.si.PMTService 502  
getServiceID() - of org.dvb.si.SIEvent 527  
getServiceID() - of org.dvb.si.SIMonitoringEvent 539  
getServiceID() - of org.dvb.si.SIService 556  
getServiceInformationType() - of org.dvb.application.storage.StoredApplicationService 1361  
getServiceLocator() - of org.dvb.application.AppAttributes 716  
getServiceType() - of org.dvb.application.storage.StoredApplicationService 1361  
getServiceType() - of org.dvb.internet.InternetClientService 1381  
getServiceXFR() - of org.dvb.dsmcc.ServiceXFRErrorEvent 675  
getServiceXFR() - of org.dvb.dsmcc.ServiceXFRException 677  
getSetupTimeEstimate() - of org.dvb.net.rc.ConnectionRCInterface 697

getShortBouquetName() - of org.dvb.si.SIBouquet 506  
getShortcutComponent(int) - of org.dvb.application.inner.DVBScene 1317  
getShortcutComponent(int) - of org.havi.ui.HScene 1026  
getShortcutKeyCode(HActionable) - of org.dvb.application.inner.DVBScene 1317  
getShortcutKeyCode(HActionable) - of org.havi.ui.HScene 1027  
getShortDescription() - of org.dvb.si.SIEvent 527  
getShortEventName() - of org.dvb.si.SIEvent 527  
getShortNetworkName() - of org.dvb.si.SINetwork 544  
getShortProviderName() - of org.dvb.si.SIService 556  
getShortServiceName() - of org.dvb.si.SIService 557  
getSIDatabase() - of org.dvb.si.SIDatabase 512  
getSIDatabase() - of org.dvb.si.SIInformation 532  
getSIDatabase(NetworkInterface) - of org.dvb.net.tuning.DvbNetworkInterfaceSIUtil 751  
getSigners() - of org.dvb.dsmcc.DSMCCObject 633  
getSIInformationType() - of org.dvb.si.SIMonitoringEvent 539  
getSIServiceLocators() - of org.dvb.si.SIBouquet 506  
getSIServiceType() - of org.dvb.si.SIService 557  
getSoftLineBreakPositions(HVisible) - of org.havi.ui.HMultilineEntryLook 985  
getSource() - of org.davic.mpeg.sections.TimeOutEvent 377  
getSource() - of org.dvb.dsmcc.AsynchronousLoadingEvent 628  
getSource() - of org.dvb.dsmcc.InvalidFormatEvent 647  
getSource() - of org.dvb.dsmcc.InvalidPathnameEvent 649  
getSource() - of org.dvb.dsmcc.LoadingAbortedEvent 651  
getSource() - of org.dvb.dsmcc.MPEGDeliveryErrorEvent 652  
getSource() - of org.dvb.dsmcc.NotEntitledEvent 654  
getSource() - of org.dvb.dsmcc.NPTRateChangeEvent 663  
getSource() - of org.dvb.dsmcc.NPTStatusEvent 665  
getSource() - of org.dvb.dsmcc.ObjectChangeEvent 666  
getSource() - of org.dvb.dsmcc.ServerDeliveryErrorEvent 669  
getSource() - of org.dvb.dsmcc.ServiceXFRErrorEvent 675  
getSource() - of org.dvb.dsmcc.StreamEvent 681  
getSource() - of org.dvb.dsmcc.SuccessEvent 683  
getSource() - of org.dvb.media.SubtitleAvailableEvent 596  
getSource() - of org.dvb.media.SubtitleNotAvailableEvent 598  
getSource() - of org.dvb.media.SubtitleNotSelectedEvent 599  
getSource() - of org.dvb.media.SubtitleSelectedEvent 600  
getSource() - of org.dvb.net.rc.RCInterfaceReleasedEvent 707  
getSource() - of org.dvb.net.rc.RCInterfaceReservedEvent 708  
getSource() - of org.dvb.si.SIMonitoringEvent 540  
getSource() - of org.dvb.si.SIRetrievalEvent 552  
getSource() - of org.dvb.ui.DVBBufferedImage 764  
getSource() - of org.havi.ui.event.HBackgroundImageEvent 1204  
getSource() - of org.havi.ui.event.HScreenDeviceReleasedEvent 1240  
getSource() - of org.havi.ui.event.HScreenDeviceReservedEvent 1241  
getSource() - of org.havi.ui.event.HScreenLocationModifiedEvent 1243  
getStartCorner() - of org.dvb.ui.DVBTextLayoutManager 779  
getStartTime() - of org.dvb.si.SIEvent 527  
getStartTime() - of org.dvb.si.SIMonitoringEvent 540  
getState() - of org.dvb.application.AppProxy 722  
getStoredAppIDs() - of org.dvb.application.storage.ApplicationStorage 1350  
getStream() - of org.dvb.media.CAStopEvent 582  
getStream() - of org.dvb.media.NoComponentSelectedEvent 589  
getStream() - of org.dvb.media.PresentationChangedEvent 591  
getStream() - of org.dvb.media.ServiceRemovedEvent 593  
getStream() - of org.dvb.media.StopByResourceLossEvent 595

getStreamLocator() - of org.dvb.dsmcc.DSMCCStream 639  
 getStreams() - of org.dvb.media.DripFeedDataSource 586  
 getStreamType() - of org.dvb.si.PMTElementaryStream 501  
 getString() - of org.havi.ui.event.HEventRepresentation 1210  
 getSubimage(int, int, int, int) - of org.dvb.ui.DVBBufferedImage 764  
 getSupportedCharacterRanges(Font) - of org.havi.ui.HFontCapabilities 894  
 getSupportedClientServices() - of org.dvb.internet.InternetClientService 1382  
 getSupportedPlugins() - of org.dvb.internet.WWWBrowserService 1394  
 getSwitchableState() - of org.havi.ui.HSwitchable 1115  
 getSwitchableState() - of org.havi.ui.HToggleButton 1148  
 getSymbol() - of org.havi.ui.event.HEventRepresentation 1210  
 getTag() - of org.dvb.si.Descriptor 495  
 getTarget() - of org.dvb.net.rc.ConnectionParameters 694  
 getTextContent(int) - of org.havi.ui.HSinglelineEntry 1073  
 getTextContent(int) - of org.havi.ui.HVisible 1188  
 getLayoutManager() - of org.havi.ui.HVisible 1188  
 getTextualServiceIdentifiers() - of org.dvb.si.SIService 557  
 getTextualServiceIdentifiers() - of org.dvb.si.TextualServiceIdentifierQuery 573  
 getTextWrapping() - of org.dvb.ui.DVBTextLayoutManager 779  
 getThumbMaxOffset() - of org.havi.ui.HStaticRange 1103  
 getThumbMinOffset() - of org.havi.ui.HStaticRange 1103  
 getToggleGroup() - of org.havi.ui.HToggleButton 1148  
 getToState() - of org.dvb.application.AppStateChangeEvent 737  
 getTotalVideoArea() - of org.dvb.media.VideoPresentationControl 611  
 getTotalVideoAreaOnScreen() - of org.dvb.media.VideoPresentationControl 611  
 getTransferId() - of org.havi.ui.event.HFocusEvent 1214  
 getTransportStreamID() - of org.dvb.si.PMTElementaryStream 501  
 getTransportStreamID() - of org.dvb.si.PMTService 503  
 getTransportStreamID() - of org.dvb.si.SIEvent 527  
 getTransportStreamID() - of org.dvb.si.SIMonitoringEvent 540  
 getTransportStreamID() - of org.dvb.si.SIService 557  
 getTransportStreamID() - of org.dvb.si.SITransportStream 567  
 getType() - of org.davic.mpeg.NotAuthorizedException 375  
 getType() - of org.dvb.application.AppAttributes 716  
 getType() - of org.dvb.event.UserEvent 464  
 getType() - of org.dvb.net.rc.RCInterface 704  
 getType() - of org.dvb.ui.DVBGraphics 773  
 getType() - of org.havi.ui.event.HEventRepresentation 1211  
 getType() - of org.havi.ui.HImageHints 934  
 getType() - of org.havi.ui.HKeyboardInputPreferred 942  
 getType() - of org.havi.ui.HSinglelineEntry 1073  
 getUndeclaredThrowable() - of org.dvb.application.plugins.UndeclaredThrowableException 1342  
 getUnitIncrement() - of org.havi.ui.HAdjustmentValue 812  
 getUnitIncrement() - of org.havi.ui.HRangeValue 1015  
 getUnsetActionSound() - of org.havi.ui.HSwitchable 1115  
 getUnsetActionSound() - of org.havi.ui.HToggleButton 1148  
 getUpdateTime() - of org.dvb.si.SIInformation 533  
 getURL() - of org.dvb.dsmcc.DSMCCObject 633  
 getUrl() - of org.dvb.internet.InternetClientEvent 1378  
 getUrl(Locator) - of org.dvb.dsmcc.ServiceDomain 673  
 getUserAgent() - of org.dvb.internet.WWWBrowserService 1394  
 getUserEmailAddress() - of org.dvb.internet.EmailClientService 1373  
 getUserEvent() - of org.dvb.event.UserEventRepository 467  
 getUsername() - of org.dvb.net.rc.ConnectionParameters 694  
 getUTCTime() - of org.dvb.si.SITime 566

getValidInput() - of org.havi.ui.HKeyboardInputPreferred 942  
 getValidInput() - of org.havi.ui.HSinglelineEntry 1073  
 getValue() - of org.havi.ui.HStaticRange 1104  
 getVersionNumber(AppID) - of org.dvb.application.storage.ApplicationStorage 1350  
 getVersions(String) - of org.dvb.application.AppAttributes 716  
 getVerticalAlign() - of org.dvb.ui.DVBTextLayoutManager 780  
 getVerticalAlignment() - of org.havi.ui.HVisible 1189  
 getVerticalScalingFactors() - of org.dvb.media.VideoPresentationControl 612  
 getVideoController() - of org.havi.ui.HVideoDevice 1171  
 getVideoDevice() - of org.havi.ui.HVideoComponent 1162  
 getVideoPosition() - of org.dvb.media.VideoTransformation 615  
 getVideoSize() - of org.dvb.media.VideoPresentationControl 612  
 getVideoSource() - of org.havi.ui.HVideoDevice 1171  
 getVideoTransformation() - of org.dvb.media.BackgroundVideoPresentationControl 579  
 getVideoTransformation(int) - of org.dvb.media.VideoFormatControl 606  
 getVisibleSoftLineBreakPositions(HVisible) - of org.havi.ui.HMultilineEntryLook 985  
 getWidth() - of org.dvb.ui.DVBBufferedImage 765  
 getWidth() - of org.havi.ui.HBackgroundImage 840  
 getWidth(ImageObserver) - of org.dvb.ui.DVBBufferedImage 765  
 getXletContainer() - of org.dvb.application.plugins.XletContainer 1343  
 goToURL(URL) - of org.dvb.internet.WWWBrowser 1392  
 GRAPHIC\_CONTENT\_CHANGE - of org.havi.ui.HVisible 1180  
 GRAPHICS\_CONFIGURATION - of org.havi.ui.HSceneTemplate 1038  
 GRAPHICS\_MIXING - of org.havi.ui.HVideoConfigTemplate 1163  
 GREEK\_EXTENDED - of org.havi.ui.HFontCapabilities 888  
 GREEK\_SYMBOLS\_AND\_COPTIC - of org.havi.ui.HFontCapabilities 888  
 group() - of org.havi.ui.HContainer 854  
 GUJARATI - of org.havi.ui.HFontCapabilities 888  
 GURMUKHI - of org.havi.ui.HFontCapabilities 889  
 h - of org.dvb.application.plugins.Proxy 1338  
 HActionable - of org.havi.ui 800  
 HActionEvent - of org.havi.ui.event 1196  
 HActionEvent(HActionable, int, String) - of org.havi.ui.event.HActionEvent 1196  
 HActionInputPreferred - of org.havi.ui 804  
 HActionListener - of org.havi.ui.event 1198  
 HAdjustableLook - of org.havi.ui 805  
 HAdjustmentEvent - of org.havi.ui.event 1199  
 HAdjustmentEvent(HAdjustmentValue, int) - of org.havi.ui.event.HAdjustmentEvent 1201  
 HAdjustmentInputPreferred - of org.havi.ui 808  
 HAdjustmentListener - of org.havi.ui.event 1202  
 HAdjustmentValue - of org.havi.ui 810  
 HALFWIDTH\_AND\_FULLWIDTH\_FORMS - of org.havi.ui.HFontCapabilities 889  
 HALIGN\_CENTER - of org.havi.ui.HVisible 1180  
 HALIGN\_JUSTIFY - of org.havi.ui.HVisible 1180  
 HALIGN\_LEFT - of org.havi.ui.HVisible 1181  
 HALIGN\_RIGHT - of org.havi.ui.HVisible 1181  
 HANGUL - of org.havi.ui.HFontCapabilities 889  
 HANGUL\_COMPATIBILITY\_JAMO - of org.havi.ui.HFontCapabilities 889  
 HANGUL\_JAMO - of org.havi.ui.HFontCapabilities 889  
 HANGUL\_SUPPLEMENTARY\_A - of org.havi.ui.HFontCapabilities 889  
 HANGUL\_SUPPLEMENTARY\_B - of org.havi.ui.HFontCapabilities 889  
 HAnimateEffect - of org.havi.ui 814  
 HAnimateLook - of org.havi.ui 818  
 HAnimateLook() - of org.havi.ui.HAnimateLook 819  
 HAnimation - of org.havi.ui 823

HAnimation() - of org.havi.ui.HAnimation 825  
HAnimation(Image[], Image[], int, int, int) - of org.havi.ui.HAnimation 825  
HAnimation(Image[], Image[], int, int, int, int, int, int, int) - of org.havi.ui.HAnimation 825  
HAnimation(Image[], int, int, int) - of org.havi.ui.HAnimation 826  
HAnimation(Image[], int, int, int, int, int, int, int) - of org.havi.ui.HAnimation 826  
hasFailed() - of org.dvb.application.AppStateChangeEvent 737  
hashCode() - of org.dvb.application.AppsControlPermission 726  
hasMultipleInstances() - of org.dvb.application.storage.StoredApplicationService 1361  
hasMultipleInstances() - of org.dvb.internet.InternetClientService 1382  
hasReadApplicationAccessRight() - of org.dvb.io.persistent.FileAccessPermissions 471  
hasReadOrganisationAccessRight() - of org.dvb.io.persistent.FileAccessPermissions 471  
hasReadWorldAccessRight() - of org.dvb.io.persistent.FileAccessPermissions 472  
hasValue() - of org.dvb.user.Preference 483  
hasWriteApplicationAccessRight() - of org.dvb.io.persistent.FileAccessPermissions 472  
hasWriteOrganisationAccessRight() - of org.dvb.io.persistent.FileAccessPermissions 472  
hasWriteWorldAccessRight() - of org.dvb.io.persistent.FileAccessPermissions 472  
HAVI\_IMPLEMENTATION\_NAME - of org.havi.ui.HVersion 1159  
HAVI\_IMPLEMENTATION\_VENDOR - of org.havi.ui.HVersion 1159  
HAVI\_IMPLEMENTATION\_VERSION - of org.havi.ui.HVersion 1159  
HAVI\_SPECIFICATION\_NAME - of org.havi.ui.HVersion 1159  
HAVI\_SPECIFICATION\_VENDOR - of org.havi.ui.HVersion 1159  
HAVI\_SPECIFICATION\_VERSION - of org.havi.ui.HVersion 1160  
HBackgroundConfigTemplate - of org.havi.ui 830  
HBackgroundConfigTemplate() - of org.havi.ui.HBackgroundConfigTemplate 831  
HBackgroundConfiguration - of org.havi.ui 833  
HBackgroundConfiguration() - of org.havi.ui.HBackgroundConfiguration 833  
HBackgroundDevice - of org.havi.ui 835  
HBackgroundDevice() - of org.havi.ui.HBackgroundDevice 835  
HBackgroundImage - of org.havi.ui 839  
HBackgroundImage(byte[]) - of org.havi.ui.HBackgroundImage 839  
HBackgroundImage(String) - of org.havi.ui.HBackgroundImage 839  
HBackgroundImage(URL) - of org.havi.ui.HBackgroundImage 840  
HBackgroundImageEvent - of org.havi.ui.event 1203  
HBackgroundImageEvent(Object, int) - of org.havi.ui.event.HBackgroundImageEvent 1204  
HBackgroundImageListener - of org.havi.ui.event 1206  
HChangeData - of org.havi.ui 842  
HChangeData(int, Object) - of org.havi.ui.HChangeData 842  
HComponent - of org.havi.ui 843  
HComponent() - of org.havi.ui.HComponent 844  
HComponent(int, int, int, int) - of org.havi.ui.HComponent 844  
HComponentOrdering - of org.havi.ui 846  
HConfigurationException - of org.havi.ui 850  
HConfigurationException() - of org.havi.ui.HConfigurationException 850  
HConfigurationException(String) - of org.havi.ui.HConfigurationException 850  
HContainer - of org.havi.ui 852  
HContainer() - of org.havi.ui.HContainer 853  
HContainer(int, int, int, int) - of org.havi.ui.HContainer 853  
HDefaultTextLayoutManager - of org.havi.ui 859  
HDefaultTextLayoutManager() - of org.havi.ui.HDefaultTextLayoutManager 860  
HEBREW\_EXTENDED - of org.havi.ui.HFontCapabilities 889  
height - of org.havi.ui.HScreenDimension 1059  
height - of org.havi.ui.HScreenRectangle 1063  
HEmulatedGraphicsConfiguration - of org.havi.ui 861  
HEmulatedGraphicsConfiguration() - of org.havi.ui.HEmulatedGraphicsConfiguration 861  
HEmulatedGraphicsDevice - of org.havi.ui 863

HEmulatedGraphicsDevice() - of org.havi.ui.HEmulatedGraphicsDevice 863  
HEventMulticaster - of org.havi.ui 865  
HEventMulticaster(EventListener, EventListener) - of org.havi.ui.HEventMulticaster 866  
HEventRepresentation - of org.havi.ui.event 1207  
HEventRepresentation() - of org.havi.ui.event.HEventRepresentation 1210  
HFlatEffectMatte - of org.havi.ui 877  
HFlatEffectMatte() - of org.havi.ui.HFlatEffectMatte 878  
HFlatEffectMatte(float[]) - of org.havi.ui.HFlatEffectMatte 878  
HFlatMatte - of org.havi.ui 882  
HFlatMatte() - of org.havi.ui.HFlatMatte 882  
HFlatMatte(float) - of org.havi.ui.HFlatMatte 882  
HFOCUS\_FIRST - of org.havi.ui.event.HFocusEvent 1213  
HFOCUS\_LAST - of org.havi.ui.event.HFocusEvent 1214  
HFocusEvent - of org.havi.ui.event 1213  
HFocusEvent(Component, int) - of org.havi.ui.event.HFocusEvent 1214  
HFocusEvent(Component, int, int) - of org.havi.ui.event.HFocusEvent 1214  
HFocusListener - of org.havi.ui.event 1216  
HFontCapabilities - of org.havi.ui 884  
HFontCapabilities() - of org.havi.ui.HFontCapabilities 893  
HGraphicButton - of org.havi.ui 895  
HGraphicButton() - of org.havi.ui.HGraphicButton 897  
HGraphicButton(Image) - of org.havi.ui.HGraphicButton 897  
HGraphicButton(Image, Image, Image) - of org.havi.ui.HGraphicButton 897  
HGraphicButton(Image, Image, Image, int, int, int, int) - of org.havi.ui.HGraphicButton 897  
HGraphicButton(Image, int, int, int, int) - of org.havi.ui.HGraphicButton 897  
HGraphicLook - of org.havi.ui 904  
HGraphicLook() - of org.havi.ui.HGraphicLook 905  
HGraphicsConfigTemplate - of org.havi.ui 909  
HGraphicsConfigTemplate() - of org.havi.ui.HGraphicsConfigTemplate 910  
HGraphicsConfiguration - of org.havi.ui 913  
HGraphicsConfiguration() - of org.havi.ui.HGraphicsConfiguration 913  
HGraphicsDevice - of org.havi.ui 918  
HGraphicsDevice() - of org.havi.ui.HGraphicsDevice 918  
HIcon - of org.havi.ui 922  
HIcon() - of org.havi.ui.HIcon 924  
HIcon(Image) - of org.havi.ui.HIcon 924  
HIcon(Image, Image, int, int, int, int) - of org.havi.ui.HIcon 924  
HIcon(Image, int, int, int, int) - of org.havi.ui.HIcon 924  
HImageEffectMatte - of org.havi.ui 928  
HImageEffectMatte() - of org.havi.ui.HImageEffectMatte 929  
HImageEffectMatte(Image[]) - of org.havi.ui.HImageEffectMatte 929  
HImageHints - of org.havi.ui 933  
HImageHints() - of org.havi.ui.HImageHints 934  
HImageMatte - of org.havi.ui 935  
HImageMatte() - of org.havi.ui.HImageMatte 935  
HImageMatte(Image) - of org.havi.ui.HImageMatte 935  
hint - of org.havi.ui.HChangeData 842  
HInvalidLookException - of org.havi.ui 937  
HInvalidLookException() - of org.havi.ui.HInvalidLookException 937  
HInvalidLookException(String) - of org.havi.ui.HInvalidLookException 937  
HIRAGANA - of org.havi.ui.HFontCapabilities 890  
HItemEvent - of org.havi.ui.event 1217  
HItemEvent(HItemValue, int, Object) - of org.havi.ui.event.HItemEvent 1219  
HItemListener - of org.havi.ui.event 1220  
HItemValue - of org.havi.ui 938



hitTest(HAdjustmentValue, Point) - of org.havi.ui.HAdjustableLook 807  
hitTest(HAdjustmentValue, Point) - of org.havi.ui.HListGroupLook 965  
hitTest(HAdjustmentValue, Point) - of org.havi.ui.HRangeLook 1009  
HKeyboardInputPreferred - of org.havi.ui 941  
HKeyCapabilities - of org.havi.ui.event 1221  
HKeyCapabilities() - of org.havi.ui.event.HKeyCapabilities 1221  
HKeyEvent - of org.havi.ui.event 1223  
HKeyEvent(Component, int, long, int, int) - of org.havi.ui.event.HKeyEvent 1224  
HKeyEvent(Component, int, long, int, int, char) - of org.havi.ui.event.HKeyEvent 1224  
HKeyListener - of org.havi.ui.event 1225  
HListElement - of org.havi.ui 944  
HListElement(Image, String) - of org.havi.ui.HListElement 944  
HListElement(String) - of org.havi.ui.HListElement 945  
HListGroup - of org.havi.ui 946  
HListGroup() - of org.havi.ui.HListGroup 949  
HListGroup(HListElement[]) - of org.havi.ui.HListGroup 949  
HListGroup(HListElement[], int, int, int, int) - of org.havi.ui.HListGroup 949  
HListGroupLook - of org.havi.ui 961  
HListGroupLook() - of org.havi.ui.HListGroupLook 962  
HLook - of org.havi.ui 967  
HMatte - of org.havi.ui 973  
HMatteException - of org.havi.ui 974  
HMatteException() - of org.havi.ui.HMatteException 974  
HMatteException(String) - of org.havi.ui.HMatteException 975  
HMatteLayer - of org.havi.ui 976  
HMouseCapabilities - of org.havi.ui.event 1226  
HMouseCapabilities() - of org.havi.ui.event.HMouseCapabilities 1226  
HMultilineEntry - of org.havi.ui 977  
HMultilineEntry() - of org.havi.ui.HMultilineEntry 979  
HMultilineEntry(int) - of org.havi.ui.HMultilineEntry 979  
HMultilineEntry(int, int, int, int, int) - of org.havi.ui.HMultilineEntry 979  
HMultilineEntry(String, int, Font, Color) - of org.havi.ui.HMultilineEntry 979  
HMultilineEntry(String, int, int, int, int, int, Font, Color) - of org.havi.ui.HMultilineEntry 979  
HMultilineEntryLook - of org.havi.ui 981  
HMultilineEntryLook() - of org.havi.ui.HMultilineEntryLook 981  
HNavigable - of org.havi.ui 987  
HNavigationInputPreferred - of org.havi.ui 992  
HNoInputPreferred - of org.havi.ui 994  
HomePagePermission - of org.dvb.internet 1375  
HomePagePermission(String) - of org.dvb.internet.HomePagePermission 1375  
HomePagePermission(String, String) - of org.dvb.internet.HomePagePermission 1375  
HOrientable - of org.havi.ui 995  
HORIZONTAL\_CENTER - of org.dvb.ui.DVBTextLayoutManager 776  
HORIZONTAL\_END\_ALIGN - of org.dvb.ui.DVBTextLayoutManager 776  
HORIZONTAL\_START\_ALIGN - of org.dvb.ui.DVBTextLayoutManager 776  
HPermissionDeniedException - of org.havi.ui 997  
HPermissionDeniedException() - of org.havi.ui.HPermissionDeniedException 997  
HPermissionDeniedException(String) - of org.havi.ui.HPermissionDeniedException 998  
HRange - of org.havi.ui 999  
HRange() - of org.havi.ui.HRange 1001  
HRange(int, int, int, int) - of org.havi.ui.HRange 1001  
HRange(int, int, int, int, int, int, int, int) - of org.havi.ui.HRange 1001  
HRangeLook - of org.havi.ui 1006  
HRangeLook() - of org.havi.ui.HRangeLook 1006  
HRangeValue - of org.havi.ui 1011

HRangeValue() - of org.havi.ui.HRangeValue 1013  
HRangeValue(int, int, int, int) - of org.havi.ui.HRangeValue 1013  
HRangeValue(int, int, int, int, int, int, int, int) - of org.havi.ui.HRangeValue 1013  
HRcCapabilities - of org.havi.ui.event 1227  
HRcCapabilities() - of org.havi.ui.event.HRcCapabilities 1227  
HRcEvent - of org.havi.ui.event 1229  
HRcEvent(Component, int, long, int, int) - of org.havi.ui.event.HRcEvent 1236  
HRcEvent(Component, int, long, int, int, char) - of org.havi.ui.event.HRcEvent 1237  
HScene - of org.havi.ui 1020  
HScene() - of org.havi.ui.HScene 1023  
HSceneFactory - of org.havi.ui 1033  
HSceneTemplate - of org.havi.ui 1037  
HSceneTemplate() - of org.havi.ui.HSceneTemplate 1039  
HScreen - of org.havi.ui 1041  
HScreenConfigTemplate - of org.havi.ui 1046  
HScreenConfigTemplate() - of org.havi.ui.HScreenConfigTemplate 1050  
HScreenConfiguration - of org.havi.ui 1052  
HScreenConfigurationEvent - of org.havi.ui.event 1238  
HScreenConfigurationEvent(Object) - of org.havi.ui.event.HScreenConfigurationEvent 1238  
HScreenConfigurationListener - of org.havi.ui.event 1239  
HScreenDevice - of org.havi.ui 1054  
HScreenDeviceReleasedEvent - of org.havi.ui.event 1240  
HScreenDeviceReleasedEvent(Object) - of org.havi.ui.event.HScreenDeviceReleasedEvent 1240  
HScreenDeviceReservedEvent - of org.havi.ui.event 1241  
HScreenDeviceReservedEvent(Object) - of org.havi.ui.event.HScreenDeviceReservedEvent 1241  
HScreenDimension - of org.havi.ui 1058  
HScreenDimension(float, float) - of org.havi.ui.HScreenDimension 1059  
HScreenLocationModifiedEvent - of org.havi.ui.event 1242  
HScreenLocationModifiedEvent(Object) - of org.havi.ui.event.HScreenLocationModifiedEvent 1242  
HScreenLocationModifiedListener - of org.havi.ui.event 1244  
HScreenPoint - of org.havi.ui 1060  
HScreenPoint(float, float) - of org.havi.ui.HScreenPoint 1061  
HScreenRectangle - of org.havi.ui 1062  
HScreenRectangle(float, float, float, float) - of org.havi.ui.HScreenRectangle 1063  
HSelectionInputPreferred - of org.havi.ui 1065  
HSinglelineEntry - of org.havi.ui 1067  
HSinglelineEntry() - of org.havi.ui.HSinglelineEntry 1069  
HSinglelineEntry(int) - of org.havi.ui.HSinglelineEntry 1070  
HSinglelineEntry(int, int, int, int, int) - of org.havi.ui.HSinglelineEntry 1070  
HSinglelineEntry(String, int, Font, Color) - of org.havi.ui.HSinglelineEntry 1070  
HSinglelineEntry(String, int, int, int, int, int, Font, Color) - of org.havi.ui.HSinglelineEntry 1070  
HSinglelineEntryLook - of org.havi.ui 1079  
HSinglelineEntryLook() - of org.havi.ui.HSinglelineEntryLook 1079  
HSound - of org.havi.ui 1084  
HSound() - of org.havi.ui.HSound 1084  
HState - of org.havi.ui 1087  
HStaticAnimation - of org.havi.ui 1091  
HStaticAnimation() - of org.havi.ui.HStaticAnimation 1093  
HStaticAnimation(Image[], int, int, int) - of org.havi.ui.HStaticAnimation 1093  
HStaticAnimation(Image[], int, int, int, int, int, int, int) - of org.havi.ui.HStaticAnimation 1093  
HStaticIcon - of org.havi.ui 1097  
HStaticIcon() - of org.havi.ui.HStaticIcon 1098  
HStaticIcon(Image) - of org.havi.ui.HStaticIcon 1098  
HStaticIcon(Image, int, int, int, int) - of org.havi.ui.HStaticIcon 1099  
HStaticRange - of org.havi.ui 1100

HStaticRange() - of org.havi.ui.HStaticRange 1102  
 HStaticRange(int, int, int, int) - of org.havi.ui.HStaticRange 1102  
 HStaticRange(int, int, int, int, int, int, int, int) - of org.havi.ui.HStaticRange 1102  
 HStaticText - of org.havi.ui 1106  
 HStaticText() - of org.havi.ui.HStaticText 1108  
 HStaticText(String) - of org.havi.ui.HStaticText 1108  
 HStaticText(String, Font, Color, Color, HTextLayoutManager) - of org.havi.ui.HStaticText 1108  
 HStaticText(String, int, int, int, int) - of org.havi.ui.HStaticText 1108  
 HStaticText(String, int, int, int, int, Font, Color, Color, HTextLayoutManager) - of org.havi.ui.HStaticText 1108  
 HStillImageBackgroundConfiguration - of org.havi.ui 1110  
 HStillImageBackgroundConfiguration() - of org.havi.ui.HStillImageBackgroundConfiguration 1110  
 HSwitchable - of org.havi.ui 1113  
 HText - of org.havi.ui 1116  
 HText() - of org.havi.ui.HText 1118  
 HText(String) - of org.havi.ui.HText 1118  
 HText(String, Font, Color, Color, HTextLayoutManager) - of org.havi.ui.HText 1118  
 HText(String, int, int, int, int) - of org.havi.ui.HText 1118  
 HText(String, int, int, int, int, Font, Color, Color, HTextLayoutManager) - of org.havi.ui.HText 1119  
 HText(String, String) - of org.havi.ui.HText 1119  
 HText(String, String, Font, Color, Color, HTextLayoutManager) - of org.havi.ui.HText 1119  
 HText(String, String, int, int, int, int) - of org.havi.ui.HText 1119  
 HText(String, String, int, int, int, int, Font, Color, Color, HTextLayoutManager) - of org.havi.ui.HText 1119  
 HTextButton - of org.havi.ui 1123  
 HTextButton() - of org.havi.ui.HTextButton 1125  
 HTextButton(String) - of org.havi.ui.HTextButton 1125  
 HTextButton(String, Font, Color, Color, HTextLayoutManager) - of org.havi.ui.HTextButton 1125  
 HTextButton(String, int, int, int, int) - of org.havi.ui.HTextButton 1126  
 HTextButton(String, int, int, int, int, Font, Color, Color, HTextLayoutManager) - of org.havi.ui.HTextButton 1126  
 HTextEvent - of org.havi.ui.event 1245  
 HTextEvent(HTextValue, int) - of org.havi.ui.event.HTextEvent 1248  
 HTextLayoutManager - of org.havi.ui 1132  
 HTextListener - of org.havi.ui.event 1249  
 HTextLook - of org.havi.ui 1135  
 HTextLook() - of org.havi.ui.HTextLook 1136  
 HTextValue - of org.havi.ui 1140  
 HTMLApplication - of org.dvb.dom.inner 1312  
 HTMLApplication(URL, String, String) - of org.dvb.dom.inner.HTMLApplication 1312  
 HTMLApplication(URL, String, String, String[], String[]) - of org.dvb.dom.inner.HTMLApplication 1312  
 HTMLInnerApplicationContainer - of org.dvb.dom.inner 1313  
 HTMLInnerApplicationContainer(HTMLApplication) - of org.dvb.dom.inner.HTMLInnerApplicationContainer 1313  
 HToggleButton - of org.havi.ui 1142  
 HToggleButton() - of org.havi.ui.HToggleButton 1144  
 HToggleButton(Image) - of org.havi.ui.HToggleButton 1144  
 HToggleButton(Image, boolean, HToggleGroup) - of org.havi.ui.HToggleButton 1144  
 HToggleButton(Image, Image, Image, Image, boolean) - of org.havi.ui.HToggleButton 1145  
 HToggleButton(Image, Image, Image, Image, boolean, HToggleGroup) - of org.havi.ui.HToggleButton 1145  
 HToggleButton(Image, Image, Image, Image, int, int, int, int, boolean) - of org.havi.ui.HToggleButton 1145  
 HToggleButton(Image, Image, Image, Image, int, int, int, int, boolean, HToggleGroup) - of org.havi.ui.HToggleButton 1145  
 HToggleButton(Image, int, int, int, int) - of org.havi.ui.HToggleButton 1145  
 HToggleButton(Image, int, int, int, int, boolean) - of org.havi.ui.HToggleButton 1145  
 HToggleButton(Image, int, int, int, int, boolean, HToggleGroup) - of org.havi.ui.HToggleButton 1145

HToggleGroup - of org.havi.ui 1153  
 HToggleGroup() - of org.havi.ui.HToggleGroup 1154  
 HUIException - of org.havi.ui 1157  
 HUIException() - of org.havi.ui.HUIException 1157  
 HUIException(String) - of org.havi.ui.HUIException 1158  
 HUMAN\_INTERVENTION - of org.dvb.test.DVBTest 1255  
 HVersion - of org.havi.ui 1159  
 HVideoComponent - of org.havi.ui 1161  
 HVideoComponent() - of org.havi.ui.HVideoComponent 1161  
 HVideoConfigTemplate - of org.havi.ui 1163  
 HVideoConfigTemplate() - of org.havi.ui.HVideoConfigTemplate 1164  
 HVideoConfiguration - of org.havi.ui 1166  
 HVideoConfiguration() - of org.havi.ui.HVideoConfiguration 1166  
 HVideoDevice - of org.havi.ui 1168  
 HVideoDevice() - of org.havi.ui.HVideoDevice 1169  
 HVisible - of org.havi.ui 1173  
 HVisible() - of org.havi.ui.HVisible 1184  
 HVisible(HLook) - of org.havi.ui.HVisible 1185  
 HVisible(HLook, int, int, int, int) - of org.havi.ui.HVisible 1185  
 IllegalObjectTypeException - of org.dvb.dsmcc 644  
 IllegalObjectTypeException() - of org.dvb.dsmcc.IllegalObjectTypeException 644  
 IllegalObjectTypeException(String) - of org.dvb.dsmcc.IllegalObjectTypeException 644  
 IllegalProfileParameterException - of org.dvb.application 744  
 IllegalProfileParameterException() - of org.dvb.application.IllegalProfileParameterException 744  
 IllegalProfileParameterException(String) - of org.dvb.application.IllegalProfileParameterException 744  
 IMAGE\_CENTER - of org.havi.ui.HScene 1022  
 IMAGE\_NONE - of org.havi.ui.HScene 1022  
 IMAGE\_SCALING\_SUPPORT - of org.havi.ui.HGraphicsConfigTemplate 909  
 IMAGE\_STRETCH - of org.havi.ui.HScene 1022  
 IMAGE\_TILE - of org.havi.ui.HScene 1023  
 imageLoaded(HBackgroundImageEvent) - of org.havi.ui.event.HBackgroundImageListener 1206  
 imageLoaded(HBackgroundImageEvent) - of org.havi.ui.HEventMulticaster 870  
 imageLoadFailed(HBackgroundImageEvent) - of org.havi.ui.event.HBackgroundImageListener 1206  
 imageLoadFailed(HBackgroundImageEvent) - of org.havi.ui.HEventMulticaster 870  
 implies(Permission) - of org.dvb.application.AppsControlPermission 726  
 IncompleteTargetException - of org.dvb.net.rc 701  
 IncompleteTargetException() - of org.dvb.net.rc.IncompleteTargetException 701  
 IncompleteTargetException(String) - of org.dvb.net.rc.IncompleteTargetException 701  
 init() - of org.dvb.application.DVBProxy 742  
 initApplication(AppAttributes) - of org.dvb.application.plugins.Plugin 1334  
 initApplication(InnerApplication) - of org.dvb.application.plugins.Plugin 1334  
 initiateApplicationInstall(AppProxy, boolean, String[]) - of org.dvb.application.storage.ApplicationStorage 1351  
 initiateApplicationRemoval(AppID) - of org.dvb.application.storage.ApplicationStorage 1352  
 initPlugin() - of org.dvb.application.plugins.Plugin 1335  
 InnerApplication - of org.dvb.application.inner 1320  
 InnerApplication() - of org.dvb.application.inner.InnerApplication 1320  
 InnerApplicationContainer - of org.dvb.application.inner 1321  
 InnerApplicationContainer(InnerApplication) - of org.dvb.application.inner.InnerApplicationContainer 1321  
 InnerApplicationEvent - of org.dvb.application.inner 1326  
 InnerApplicationEvent(InnerApplicationContainer) - of org.dvb.application.inner.InnerApplicationEvent 1326  
 InnerApplicationExited(InnerApplicationEvent) - of org.dvb.application.inner.InnerApplicationListener 1327  
 InnerApplicationListener - of org.dvb.application.inner 1327  
 INPUT\_ALPHA - of org.havi.ui.HKeyboardInputPreferred 941

INPUT\_ALPHANUMERIC - of org.havi.ui.HKeyboardInputPreferred 941  
INPUT\_ANY - of org.havi.ui.HKeyboardInputPreferred 941  
INPUT\_CUSTOMIZED - of org.havi.ui.HKeyboardInputPreferred 942  
INPUT\_NUMERIC - of org.havi.ui.HKeyboardInputPreferred 942  
insertChar(char) - of org.havi.ui.HSinglelineEntry 1074  
InsufficientResourcesException - of org.dvb.dsmcc 645  
InsufficientResourcesException() - of org.dvb.dsmcc.InsufficientResourcesException 645  
InsufficientResourcesException(String) - of org.dvb.dsmcc.InsufficientResourcesException 645  
INTERLACED\_DISPLAY - of org.havi.ui.HScreenConfigTemplate 1047  
INTERNET\_CLIENT - of org.dvb.internet.InternetServiceType 1386  
InternetClient - of org.dvb.internet 1376  
InternetClientEvent - of org.dvb.internet 1378  
InternetClientEvent(Object, URL) - of org.dvb.internet.InternetClientEvent 1378  
InternetClientFailureEvent - of org.dvb.internet 1379  
InternetClientFailureEvent(Object, URL) - of org.dvb.internet.InternetClientFailureEvent 1379  
InternetClientListener - of org.dvb.internet 1380  
InternetClientService - of org.dvb.internet 1381  
InternetClientSuccessEvent - of org.dvb.internet 1383  
InternetClientSuccessEvent(Object, URL) - of org.dvb.internet.InternetClientSuccessEvent 1383  
InternetServiceFilter - of org.dvb.internet 1384  
InternetServiceFilter(int) - of org.dvb.internet.InternetServiceFilter 1384  
InternetServiceType - of org.dvb.internet 1386  
InternetServiceType(String) - of org.dvb.internet.InternetServiceType 1386  
InvalidAddressException - of org.dvb.dsmcc 646  
InvalidAddressException() - of org.dvb.dsmcc.InvalidAddressException 646  
InvalidAddressException(String) - of org.dvb.dsmcc.InvalidAddressException 646  
InvalidApplicationException - of org.dvb.application.plugins 1332  
InvalidApplicationException - of org.dvb.application.storage 1356  
InvalidApplicationException() - of org.dvb.application.plugins.InvalidApplicationException 1332  
InvalidApplicationException() - of org.dvb.application.storage.InvalidApplicationException 1356  
InvalidApplicationException(String) - of org.dvb.application.plugins.InvalidApplicationException 1332  
InvalidApplicationException(String) - of org.dvb.application.storage.InvalidApplicationException 1356  
InvalidDescriptionFileException - of org.dvb.application.storage 1357  
InvalidDescriptionFileException() - of org.dvb.application.storage.InvalidDescriptionFileException 1357  
InvalidDescriptionFileException(String) - of org.dvb.application.storage.InvalidDescriptionFileException 1357  
InvalidFormatEvent - of org.dvb.dsmcc 647  
InvalidFormatEvent(DSMCCObject) - of org.dvb.dsmcc.InvalidFormatEvent 647  
InvalidFormatException - of org.dvb.dsmcc 648  
InvalidFormatException() - of org.dvb.dsmcc.InvalidFormatException 648  
InvalidFormatException(String) - of org.dvb.dsmcc.InvalidFormatException 648  
InvalidLocatorException - of org.davic.net 379  
InvalidLocatorException() - of org.davic.net.InvalidLocatorException 379  
InvalidLocatorException(String) - of org.davic.net.InvalidLocatorException 379  
InvalidPathnameEvent - of org.dvb.dsmcc 649  
InvalidPathnameEvent(DSMCCObject) - of org.dvb.dsmcc.InvalidPathnameEvent 649  
InvalidPathNameException - of org.dvb.dsmcc 650  
InvalidPathNameException() - of org.dvb.dsmcc.InvalidPathNameException 650  
InvalidPathNameException(String) - of org.dvb.dsmcc.InvalidPathNameException 650  
InvocationHandler - of org.dvb.application.plugins 1333  
invoke(Object, Method, Object[]) - of org.dvb.application.plugins.InvocationHandler 1333  
IPA\_EXTENSIONS - of org.havi.ui.HFontCapabilities 890  
isAltDown() - of org.dvb.event.UserEvent 464  
isAnimated() - of org.havi.ui.HAnimateEffect 816  
isAnimated() - of org.havi.ui.HFlatEffectMatte 879

isAnimated() - of org.havi.ui.HImageEffectMatte 930  
 isAnimated() - of org.havi.ui.HStaticAnimation 1094  
 isAttached() - of org.dvb.dsmcc.ServiceDomain 674  
 isAudio() - of org.dvb.dsmcc.DSMCCStream 639  
 isAvailableInCache() - of org.dvb.si.SIRequest 549  
 isCharAvailable(Font, char) - of org.havi.ui.HFontCapabilities 894  
 isConfigSupported(HBackgroundConfiguration) - of org.havi.ui.HBackgroundConfigTemplate 832  
 isConfigSupported(HGraphicsConfiguration) - of org.havi.ui.HGraphicsConfigTemplate 911  
 isConfigSupported(HVideoConfiguration) - of org.havi.ui.HVideoConfigTemplate 1165  
 isConnected() - of org.dvb.net.rc.ConnectionRCInterface 697  
 isControlDown() - of org.dvb.event.UserEvent 464  
 isData() - of org.dvb.dsmcc.DSMCCStream 639  
 isDoubleBuffered() - of org.dvb.application.inner.DVBScene 1318  
 isDoubleBuffered() - of org.havi.ui.HComponent 844  
 isDoubleBuffered() - of org.havi.ui.HContainer 854  
 isDoubleBuffered() - of org.havi.ui.HScene 1027  
 isEnabled() - of org.havi.ui.HToggleGroup 1154  
 isEnabledShortcuts() - of org.dvb.application.inner.DVBScene 1318  
 isEnabledShortcuts() - of org.havi.ui.HScene 1027  
 isFocusTraversable() - of org.havi.ui.HVisible 1189  
 isGrouped() - of org.havi.ui.HContainer 855  
 isSelected(int) - of org.havi.ui.HListGroup 954  
 isLoaded() - of org.dvb.dsmcc.DSMCCObject 634  
 isMetaDown() - of org.dvb.event.UserEvent 464  
 isMPEGProgram() - of org.dvb.dsmcc.DSMCCStream 639  
 isNetworkConnectionAvailable() - of org.dvb.dsmcc.ServiceDomain 674  
 isObjectKindKnown() - of org.dvb.dsmcc.DSMCCObject 634  
 isOpaque() - of org.dvb.application.inner.DVBScene 1318  
 isOpaque() - of org.dvb.ui.TestOpacity 787  
 isOpaque() - of org.havi.ui.HComponent 844  
 isOpaque() - of org.havi.ui.HContainer 855  
 isOpaque() - of org.havi.ui.HScene 1027  
 isOpaque() - of org.havi.ui.HVisible 1189  
 isOpaque(HVisible) - of org.havi.ui.HAnimateLook 821  
 isOpaque(HVisible) - of org.havi.ui.HGraphicLook 907  
 isOpaque(HVisible) - of org.havi.ui.HListGroupLook 965  
 isOpaque(HVisible) - of org.havi.ui.HLook 971  
 isOpaque(HVisible) - of org.havi.ui.HMultilineEntryLook 985  
 isOpaque(HVisible) - of org.havi.ui.HRangeLook 1009  
 isOpaque(HVisible) - of org.havi.ui.HSinglelineEntryLook 1082  
 isOpaque(HVisible) - of org.havi.ui.HTextLook 1138  
 isPanAndScan() - of org.dvb.media.VideoTransformation 615  
 isProxyClass(Class) - of org.dvb.application.plugins.Proxy 1339  
 isSelected() - of org.dvb.application.inner.InnerApplicationContainer 1323  
 isSelected() - of org.havi.ui.HAnimation 827  
 isSelected() - of org.havi.ui.HGraphicButton 899  
 isSelected() - of org.havi.ui.HIcon 925  
 isSelected() - of org.havi.ui.HListGroup 955  
 isSelected() - of org.havi.ui.HNavigable 989  
 isSelected() - of org.havi.ui.HRange 1002  
 isSelected() - of org.havi.ui.HRangeValue 1016  
 isSelected() - of org.havi.ui.HSinglelineEntry 1074  
 isSelected() - of org.havi.ui.HText 1120  
 isSelected() - of org.havi.ui.HTextButton 1128  
 isSelected() - of org.havi.ui.HToggleButton 1148

isShiftDown() - of org.dvb.event.UserEvent 464  
 isStartable() - of org.dvb.application.AppAttributes 716  
 isStream() - of org.dvb.dsmcc.DSMCCObject 634  
 isStreamEvent() - of org.dvb.dsmcc.DSMCCObject 634  
 isSupported() - of org.havi.ui.event.HEventRepresentation 1211  
 isSupported(AppAttributes) - of org.dvb.application.plugins.Plugin 1335  
 isSupported(int) - of org.havi.ui.event.HKeyCapabilities 1222  
 isSupported(int) - of org.havi.ui.event.HRcCapabilities 1228  
 isTemporary() - of org.havi.ui.event.HFocusEvent 1215  
 isVideo() - of org.dvb.dsmcc.DSMCCStream 639  
 isVisible() - of org.dvb.application.inner.DVBScene 1318  
 isVisible() - of org.havi.ui.HScene 1028  
 ITEM\_CLEARED - of org.havi.ui.event.HItemEvent 1217  
 ITEM\_END\_CHANGE - of org.havi.ui.event.HItemEvent 1217  
 ITEM\_FIRST - of org.havi.ui.event.HItemEvent 1218  
 ITEM\_LAST - of org.havi.ui.event.HItemEvent 1218  
 ITEM\_NOT\_FOUND - of org.havi.ui.HListGroup 949  
 ITEM\_SELECTED - of org.havi.ui.event.HItemEvent 1218  
 ITEM\_SELECTION\_CLEARED - of org.havi.ui.event.HItemEvent 1218  
 ITEM\_SET\_CURRENT - of org.havi.ui.event.HItemEvent 1218  
 ITEM\_SET\_NEXT - of org.havi.ui.event.HItemEvent 1218  
 ITEM\_SET\_PREVIOUS - of org.havi.ui.event.HItemEvent 1218  
 ITEM\_START\_CHANGE - of org.havi.ui.event.HItemEvent 1218  
 ITEM\_TOGGLE\_SELECTED - of org.havi.ui.event.HItemEvent 1219  
 ITEM\_VALUE\_CHANGE - of org.havi.ui.HVisible 1181  
 IxcRegistry - of org.dvb.io.ixc 1263  
 KANNADA - of org.havi.ui.HFontCapabilities 890  
 KATAKANA - of org.havi.ui.HFontCapabilities 890  
 keyPressed(KeyEvent) - of org.havi.ui.HEventMulticaster 870  
 keyReleased(KeyEvent) - of org.havi.ui.HEventMulticaster 871  
 keyTyped(KeyEvent) - of org.havi.ui.HEventMulticaster 871  
 LanguageNotAvailableException - of org.dvb.application 745  
 LanguageNotAvailableException() - of org.dvb.application.LanguageNotAvailableException 745  
 LanguageNotAvailableException(String) - of org.dvb.application.LanguageNotAvailableException 745  
 LAO - of org.havi.ui.HFontCapabilities 890  
 LARGEST\_PIXEL\_DIMENSION - of org.havi.ui.HSceneTemplate 1038  
 LAST\_CHANGE - of org.havi.ui.HVisible 1181  
 LAST\_STATE - of org.havi.ui.HState 1090  
 LATIN\_1\_SUPPLEMENT - of org.havi.ui.HFontCapabilities 890  
 LATIN\_EXTENDED\_A - of org.havi.ui.HFontCapabilities 890  
 LATIN\_EXTENDED\_ADDITIONAL - of org.havi.ui.HFontCapabilities 890  
 LATIN\_EXTENDED\_B - of org.havi.ui.HFontCapabilities 891  
 LETTERLIKE\_SYMBOLS - of org.havi.ui.HFontCapabilities 891  
 LINE\_ART - of org.havi.ui.HImageHints 933  
 LINE\_ORIENTATION\_HORIZONTAL - of org.dvb.ui.DVBTextLayoutManager 776  
 LINE\_ORIENTATION\_VERTICAL - of org.dvb.ui.DVBTextLayoutManager 777  
 LINKAGE - of org.dvb.si.DescriptorTag 497  
 list(XletContext) - of org.dvb.io.ixc.IxcRegistry 1264  
 LIST\_CONTENT\_CHANGE - of org.havi.ui.HVisible 1181  
 LIST\_ICONSIZING\_CHANGE - of org.havi.ui.HVisible 1181  
 LIST\_LABELSIZING\_CHANGE - of org.havi.ui.HVisible 1181  
 LIST\_MULTISELECTION\_CHANGE - of org.havi.ui.HVisible 1182  
 LIST\_SCROLLPOSITION\_CHANGE - of org.havi.ui.HVisible 1182  
 load() - of org.dvb.application.DVBProxy 742  
 load(HBackgroundImageListener) - of org.havi.ui.HBackgroundImage 841

load(String) - of org.havi.ui.HSound 1084  
 load(URL) - of org.havi.ui.HSound 1085  
 loadDirectoryEntry(AsynchronousLoadingEventListener) - of org.dvb.dsmcc.DSMCCObject 634  
 LOADED - of org.dvb.application.DVBProxy 742  
 LoadingAbortedEvent - of org.dvb.dsmcc 651  
 LoadingAbortedEvent(DSMCCObject) - of org.dvb.dsmcc.LoadingAbortedEvent 651  
 LOCAL\_TIME\_OFFSET - of org.dvb.si.DescriptorTag 497  
 log(String, int) - of org.dvb.test.DVBTest 1256  
 log(String, String) - of org.dvb.test.DVBTest 1256  
 lookup(XletContext, String) - of org.dvb.io.ixc.IxcRegistry 1264  
 loop() - of org.havi.ui.HSound 1085  
 MALAYALAM - of org.havi.ui.HFontCapabilities 891  
 MATHEMATICAL\_OPERATORS - of org.havi.ui.HFontCapabilities 891  
 MATTE\_SUPPORT - of org.havi.ui.HGraphicsConfigTemplate 909  
 MHP\_APPLICATION - of org.dvb.si.SIServiceType 561  
 MIN\_MAX\_CHANGE - of org.havi.ui.HVisible 1182  
 MISCELLANEOUS\_SYMBOLS - of org.havi.ui.HFontCapabilities 891  
 MISCELLANEOUS\_TECHNICAL - of org.havi.ui.HFontCapabilities 891  
 MMIOObject - of org.davic.net.ca 384  
 MOSAIC - of org.dvb.si.DescriptorTag 497  
 MOSAIC - of org.dvb.si.SIServiceType 562  
 MPEG1\_AUDIO - of org.dvb.si.PMTStreamType 504  
 MPEG1\_VIDEO - of org.dvb.si.PMTStreamType 504  
 MPEG2\_AUDIO - of org.dvb.si.PMTStreamType 504  
 MPEG2\_VIDEO - of org.dvb.si.PMTStreamType 504  
 MPEGDeliveryErrorEvent - of org.dvb.dsmcc 652  
 MPEGDeliveryErrorEvent(DSMCCObject) - of org.dvb.dsmcc.MPEGDeliveryErrorEvent 652  
 MPEGDeliveryException - of org.dvb.dsmcc 653  
 MPEGDeliveryException() - of org.dvb.dsmcc.MPEGDeliveryException 653  
 MPEGDeliveryException(String) - of org.dvb.dsmcc.MPEGDeliveryException 653  
 MULTILINGUAL\_BOUQUET\_NAME - of org.dvb.si.DescriptorTag 497  
 MULTILINGUAL\_COMPONENT - of org.dvb.si.DescriptorTag 497  
 MULTILINGUAL\_NETWORK\_NAME - of org.dvb.si.DescriptorTag 497  
 MULTILINGUAL\_SERVICE\_NAME - of org.dvb.si.DescriptorTag 497  
 MultipleDocumentsAction - of org.dvb.dom.bootstrap 1310  
 NATURAL\_IMAGE - of org.havi.ui.HImageHints 934  
 NETWORK - of org.dvb.si.SIMonitoringType 542  
 NETWORK\_NAME - of org.dvb.si.DescriptorTag 498  
 NEW\_DATABASE - of org.dvb.application.AppsDatabaseEvent 732  
 newDatabase(AppsDatabaseEvent) - of org.dvb.application.AppsDatabaseEventListener 734  
 newInstance(URL[]) - of org.dvb.lang.DVBClassLoader 452  
 newInstance(URL[], ClassLoader) - of org.dvb.lang.DVBClassLoader 452  
 newProxyInstance(ClassLoader, Class[], InvocationHandler) - of org.dvb.application.plugins.Proxy 1339  
 NEWS\_CLIENT - of org.dvb.internet.InternetServiceFilter 1384  
 NO\_BACKGROUND\_FILL - of org.havi.ui.HScene 1023  
 NO\_BACKGROUND\_FILL - of org.havi.ui.HVisible 1182  
 NO\_DEFAULT\_HEIGHT - of org.havi.ui.HVisible 1182  
 NO\_DEFAULT\_SIZE - of org.havi.ui.HVisible 1182  
 NO\_DEFAULT\_WIDTH - of org.havi.ui.HVisible 1182  
 NO\_TRANSFER\_ID - of org.havi.ui.event.HFocusEvent 1214  
 NoComponentSelectedEvent - of org.dvb.media 588  
 NoComponentSelectedEvent(Controller) - of org.dvb.media.NoComponentSelectedEvent 588  
 NoComponentSelectedEvent(Controller, MediaLocator) - of org.dvb.media.NoComponentSelectedEvent 588  
 NoFreeCapacityException - of org.davic.net.ca 383  
 NoFreeCapacityException() - of org.davic.net.ca.NoFreeCapacityException 383



NoFreeCapacityException(String) - of org.davic.net.ca.NoFreeCapacityException 383  
 NORMAL\_STATE - of org.havi.ui.HState 1090  
 NOT\_CONTRIBUTING - of org.havi.ui.HVideoDevice 1168  
 NOT\_LOADED - of org.dvb.application.AppProxy 721  
 NOT\_RUNNING - of org.dvb.si.SIRunningStatus 554  
 NotAuthorizedException - of org.davic.mpeg 374  
 NotAuthorizedException() - of org.davic.mpeg.NotAuthorizedException 374  
 NotAuthorizedException(String) - of org.davic.mpeg.NotAuthorizedException 374  
 NotAuthorizedMediaException(ElementaryStream[], int[]) - of org.davic.media.NotAuthorizedMediaException 378  
 NotAuthorizedMediaException(Service, int) - of org.davic.media.NotAuthorizedMediaException 378  
 NotEnoughResourcesException - of org.dvb.application.storage 1358  
 NotEnoughResourcesException() - of org.dvb.application.storage.NotEnoughResourcesException 1358  
 NotEnoughResourcesException(String) - of org.dvb.application.storage.NotEnoughResourcesException 1358  
 NotEntitledEvent - of org.dvb.dsmcc 654  
 NotEntitledEvent(DSMCCObject) - of org.dvb.dsmcc.NotEntitledEvent 654  
 NotEntitledException - of org.dvb.dsmcc 655  
 NotEntitledException() - of org.dvb.dsmcc.NotEntitledException 655  
 NotEntitledException(String) - of org.dvb.dsmcc.NotEntitledException 655  
 NothingToAbortException - of org.dvb.dsmcc 656  
 NothingToAbortException() - of org.dvb.dsmcc.NothingToAbortException 656  
 NothingToAbortException(String) - of org.dvb.dsmcc.NothingToAbortException 656  
 notifyTextOverflow(String, HVisible, boolean, boolean) - com.nokia.mhp.ui.TextOverflowListener.notifyTextOverflow(java.lang.String, org.havi.ui.HVisible, boolean, boolean) 430  
 notifyTextOverflow(String, HVisible, boolean, boolean) - of org.dvb.ui.TextOverflowListener 788  
 NotLoadedException - of org.dvb.dsmcc 657  
 NotLoadedException() - of org.dvb.dsmcc.NotLoadedException 657  
 NotLoadedException(String) - of org.dvb.dsmcc.NotLoadedException 657  
 NPTDiscontinuityEvent - of org.dvb.dsmcc 658  
 NPTDiscontinuityEvent(DSMCCStream, long, long) - of org.dvb.dsmcc.NPTDiscontinuityEvent 658  
 NPTListener - of org.dvb.dsmcc 660  
 NPTPresentEvent - of org.dvb.dsmcc 661  
 NPTPresentEvent(DSMCCStream) - of org.dvb.dsmcc.NPTPresentEvent 661  
 NPTRate - of org.dvb.dsmcc 662  
 NPTRateChangeEvent - of org.dvb.dsmcc 663  
 NPTRateChangeEvent(DSMCCStream, NPTRate) - of org.dvb.dsmcc.NPTRateChangeEvent 663  
 NPTRemovedEvent - of org.dvb.dsmcc 664  
 NPTRemovedEvent(DSMCCStream) - of org.dvb.dsmcc.NPTRemovedEvent 664  
 NPTStatusEvent - of org.dvb.dsmcc 665  
 NPTStatusEvent(DSMCCStream) - of org.dvb.dsmcc.NPTStatusEvent 665  
 NTSC - of org.dvb.si.SIServiceType 562  
 NUMBER\_FORMS - of org.havi.ui.HFontCapabilities 891  
 numberOfRemainingObjects() - of org.dvb.si.SIIterator 536

## Numer-

NVOD\_REFERENCE - of org.dvb.si.DescriptorTag 498  
 NVOD\_REFERENCE - of org.dvb.si.SIServiceType 562  
 NVOD\_TIME\_SHIFTED - of org.dvb.si.SIServiceType 562  
 ObjectChangeEvent - of org.dvb.dsmcc 666  
 ObjectChangeEvent(DSMCCObject, int) - of org.dvb.dsmcc.ObjectChangeEvent 666  
 ObjectChangeListener - of org.dvb.dsmcc 668  
 OPTICAL\_CHARACTER\_RECOGNITION - of org.havi.ui.HFontCapabilities 891  
 OPTION\_UNSUPPORTED - of org.dvb.test.DVBTest 1255  
 org.davic.net - package 379

org.dvb.application - package 712  
org.dvb.application.inner - package 1315  
org.dvb.application.plugins - package 1329  
org.dvb.application.storage - package 1348  
org.dvb.dom.bootstrap - package 1306  
org.dvb.dom.inner - package 1311  
org.dvb.dsmcc - package 626  
org.dvb.event - package 456  
org.dvb.internet - package 1366  
org.dvb.io.ixc - package 1262  
org.dvb.io.persistent - package 470  
org.dvb.lang - package 450  
org.dvb.media - package 575  
org.dvb.net - package 686  
org.dvb.net.ca - package 747  
org.dvb.net.rc - package 689  
org.dvb.net.tuning - package 750  
org.dvb.si - package 491  
org.dvb.smartcard - package 1397  
org.dvb.test - package 1253  
org.dvb.ui - package 754  
org.dvb.user - package 478  
org.havi.ui - package 795  
org.havi.ui.event - package 1194  
ORIENT\_BOTTOM\_TO\_TOP - of org.havi.ui.HOrientable 995  
ORIENT\_LEFT\_TO\_RIGHT - of org.havi.ui.HOrientable 995  
ORIENT\_RIGHT\_TO\_LEFT - of org.havi.ui.HOrientable 995  
ORIENT\_TOP\_TO\_BOTTOM - of org.havi.ui.HOrientable 995  
ORIENTATION\_CHANGE - of org.havi.ui.HVisible 1183  
ORIYA - of org.havi.ui.HFontCapabilities 892  
OverallRepository - of org.dvb.event 460  
OverallRepository() - of org.dvb.event.OverallRepository 460  
paint(Graphics) - of org.havi.ui.HScene 1028  
paint(Graphics) - of org.havi.ui.HVisible 1189  
PAL - of org.dvb.si.SIServiceType 562  
PARENTAL\_RATING - of org.dvb.si.DescriptorTag 498  
PARTIAL\_TRANSPORT\_STREAM - of org.dvb.si.DescriptorTag 498  
PASS - of org.dvb.test.DVBTest 1255  
pause() - of org.dvb.application.AppProxy 722  
PAUSED - of org.dvb.application.AppProxy 722  
PAUSING - of org.dvb.si.SIRunningStatus 554  
performAction(DocumentAction) - of org.dvb.dom.bootstrap.DocumentFactory 1308  
performAction(DocumentAction) - of org.dvb.dom.inner.HTMLInnerApplicationContainer 1313  
performActionOnFrames(String[], MultipleDocumentsAction) - of org.dvb.dom.bootstrap.DocumentFactory 1308  
performActionReadOnly(DocumentAction) - of org.dvb.dom.bootstrap.DocumentFactory 1309  
PermissionDeniedEvent - of org.dvb.internet 1387  
PermissionDeniedEvent(Object, URL) - of org.dvb.internet.PermissionDeniedEvent 1387  
PermissionDeniedException - of org.dvb.net.rc 702  
PermissionDeniedException() - of org.dvb.net.rc.PermissionDeniedException 702  
PermissionDeniedException(String) - of org.dvb.net.rc.PermissionDeniedException 702  
PIXEL\_ASPECT\_RATIO - of org.havi.ui.HScreenConfigTemplate 1047  
PIXEL\_RESOLUTION - of org.havi.ui.HScreenConfigTemplate 1047  
play() - of org.havi.ui.HSound 1085  
PLAY\_ALTERNATING - of org.havi.ui.HAnimateEffect 814

PLAY\_REPEATING - of org.havi.ui.HAnimateEffect 814  
Plugin - of org.dvb.application.plugins 1334  
PMT\_SERVICE - of org.dvb.si.SIMonitoringType 542  
PMTElementaryStream - of org.dvb.si 500  
PMTService - of org.dvb.si 502  
PMTStreamType - of org.dvb.si 504  
pop(Component) - of org.havi.ui.HComponentOrdering 847  
pop(Component) - of org.havi.ui.HContainer 855  
pop(Component) - of org.havi.ui.HScene 1028  
popInFrontOf(Component, Component) - of org.havi.ui.HComponentOrdering 847  
popInFrontOf(Component, Component) - of org.havi.ui.HContainer 856  
popInFrontOf(Component, Component) - of org.havi.ui.HScene 1028  
popToFront(Component) - of org.havi.ui.HComponentOrdering 847  
popToFront(Component) - of org.havi.ui.HContainer 856  
popToFront(Component) - of org.havi.ui.HScene 1029  
POS\_CAP\_FULL - of org.dvb.media.VideoPresentationControl 609  
POS\_CAP\_FULL\_EVEN\_LINES - of org.dvb.media.VideoPresentationControl 609  
POS\_CAP\_FULL\_EVEN\_LINES\_IF\_ENTIRE\_VIDEO\_ON\_SCREEN - of org.dvb.media.VideoPresentationControl 609  
POS\_CAP\_FULL\_IF\_ENTIRE\_VIDEO\_ON\_SCREEN - of org.dvb.media.VideoPresentationControl 610  
POS\_CAP\_OTHER - of org.dvb.media.VideoPresentationControl 610  
postMonitoringEvent(SIMonitoringEvent) - of org.dvb.si.SIMonitoringListener 541  
postRetrievalEvent(SIRetrievalEvent) - of org.dvb.si.SIRetrievalListener 553  
Preference - of org.dvb.user 481  
Preference() - of org.dvb.user.Preference 481  
Preference(String, String) - of org.dvb.user.Preference 481  
Preference(String, String[]) - of org.dvb.user.Preference 481  
PREFERRED - of org.havi.ui.HSceneTemplate 1038  
PREFERRED - of org.havi.ui.HScreenConfigTemplate 1047  
PREFERRED\_NOT - of org.havi.ui.HScreenConfigTemplate 1048  
prefetch() - of org.dvb.application.DVBHTMLProxy 740  
prefetch(DSMCCObject, String, byte) - of org.dvb.dsmcc.DSMCCObject 634  
prefetch(String, byte) - of org.dvb.dsmcc.DSMCCObject 635  
PRESENT\_FOLLOWING\_EVENT - of org.dvb.si.SIMonitoringType 542  
PresentationChangedEvent - of org.dvb.media 590  
PresentationChangedEvent(Controller, MediaLocator, int) - of org.dvb.media.PresentationChangedEvent 591  
printStackTrace() - of org.dvb.application.plugins.UndeclaredThrowableException 1342  
printStackTrace(PrintStream) - of org.dvb.application.plugins.UndeclaredThrowableException 1342  
printStackTrace(PrintWriter) - of org.dvb.application.plugins.UndeclaredThrowableException 1342  
PRIORITY\_HIGH - of org.dvb.io.persistent.FileAttributes 473  
PRIORITY\_LOW - of org.dvb.io.persistent.FileAttributes 473  
PRIORITY\_MEDIUM - of org.dvb.io.persistent.FileAttributes 473  
PRIVATE\_DATA\_SPECIFIER - of org.dvb.si.DescriptorTag 498  
PRIVATE\_USE\_AREA - of org.havi.ui.HFontCapabilities 892  
processEvent(AWTEvent) - of org.havi.ui.HComponent 845  
processHActionEvent(HActionEvent) - of org.havi.ui.HActionInputPreferred 804  
processHActionEvent(HActionEvent) - of org.havi.ui.HGraphicButton 900  
processHActionEvent(HActionEvent) - of org.havi.ui.HTextButton 1128  
processHActionEvent(HActionEvent) - of org.havi.ui.HToggleButton 1148  
processHAdjustmentEvent(HAdjustmentEvent) - of org.havi.ui.HAdjustmentInputPreferred 808  
processHAdjustmentEvent(HAdjustmentEvent) - of org.havi.ui.HRangeValue 1016  
processHFocusEvent(HFocusEvent) - of org.dvb.application.inner.InnerApplicationContainer 1323  
processHFocusEvent(HFocusEvent) - of org.havi.ui.HAnimation 827  
processHFocusEvent(HFocusEvent) - of org.havi.ui.HGraphicButton 900  
processHFocusEvent(HFocusEvent) - of org.havi.ui.HIcon 925

processHFocusEvent(HFocusEvent) - of org.havi.ui.HListGroup 955  
 processHFocusEvent(HFocusEvent) - of org.havi.ui.HNavigationInputPreferred 993  
 processHFocusEvent(HFocusEvent) - of org.havi.ui.HRange 1003  
 processHFocusEvent(HFocusEvent) - of org.havi.ui.HRangeValue 1016  
 processHFocusEvent(HFocusEvent) - of org.havi.ui.HSinglelineEntry 1074  
 processHFocusEvent(HFocusEvent) - of org.havi.ui.HText 1121  
 processHFocusEvent(HFocusEvent) - of org.havi.ui.HTextButton 1128  
 processHFocusEvent(HFocusEvent) - of org.havi.ui.HToggleButton 1149  
 processHItemEvent(HItemEvent) - of org.havi.ui.HListGroup 955  
 processHItemEvent(HItemEvent) - of org.havi.ui.HSelectionInputPreferred 1065  
 processHKeyEvent(HKeyEvent) - of org.havi.ui.HKeyboardInputPreferred 943  
 processHKeyEvent(HKeyEvent) - of org.havi.ui.HSinglelineEntry 1074  
 processHTextEvent(HTextEvent) - of org.havi.ui.HKeyboardInputPreferred 943  
 processHTextEvent(HTextEvent) - of org.havi.ui.HSinglelineEntry 1074  
 processWindowEvent(WindowEvent) - of org.havi.ui.HScene 1029  
 prompt(String, int, String) - of org.dvb.test.DVBTest 1258  
 Proxy - of org.dvb.application.plugins 1336  
 Proxy(InvocationHandler) - of org.dvb.application.plugins.Proxy 1338  
 push(Component) - of org.havi.ui.HComponentOrdering 848  
 push(Component) - of org.havi.ui.HContainer 856  
 push(Component) - of org.havi.ui.HScene 1029  
 pushBehind(Component, Component) - of org.havi.ui.HComponentOrdering 848  
 pushBehind(Component, Component) - of org.havi.ui.HContainer 857  
 pushBehind(Component, Component) - of org.havi.ui.HScene 1030  
 pushToBack(Component) - of org.havi.ui.HComponentOrdering 848  
 pushToBack(Component) - of org.havi.ui.HContainer 857  
 pushToBack(Component) - of org.havi.ui.HScene 1030  
 RC\_FIRST - of org.havi.ui.event.HRcEvent 1230  
 RC\_LAST - of org.havi.ui.event.HRcEvent 1230  
 RCInterface - of org.dvb.net.rc 703  
 RCInterfaceManager - of org.dvb.net.rc 705  
 RCInterfaceReleasedEvent - of org.dvb.net.rc 707  
 RCInterfaceReleasedEvent(Object) - of org.dvb.net.rc.RCInterfaceReleasedEvent 707  
 RCInterfaceReservedEvent - of org.dvb.net.rc 708  
 RCInterfaceReservedEvent(Object) - of org.dvb.net.rc.RCInterfaceReservedEvent 708  
 RCPermission - of org.dvb.net.rc 709  
 RCPermission(String) - of org.dvb.net.rc.RCPermission 709  
 RCPermission(String, String) - of org.dvb.net.rc.RCPermission 710  
 read(Preference) - of org.dvb.user.UserPreferenceManager 487  
 read(Preference, Facility) - of org.dvb.user.UserPreferenceManager 487  
 rebind(XletContext, String, Remote) - of org.dvb.io.ixc.IxcRegistry 1265  
 rebind(XletContext, String, Remote, boolean) - of org.dvb.io.ixc.IxcRegistry 1265  
 receiveEvent(AsynchronousLoadingEvent) - of org.dvb.dsmcc.AsynchronousLoadingEventListener 629  
 ReceiveInternetClientEvent(InternetClientEvent) - of org.dvb.internet.InternetClientListener 1380  
 receiveNPTStatusEvent(NPTStatusEvent) - of org.dvb.dsmcc.NPTListener 660  
 receiveObjectChangeEvent(ObjectChangeEvent) - of org.dvb.dsmcc.ObjectChangeEventListener 668  
 receiveRateChangedEvent(NPTRateChangeEvent) - of org.dvb.dsmcc.NPTListener 660  
 receiveStreamEvent(StreamEvent) - of org.dvb.dsmcc.StreamEventListener 682  
 receiveUserPreferenceChangeEvent(UserPreferenceChangeEvent) - of org.dvb.user.UserPreferenceChangeListener 486  
 receiveVideoFormatEvent(VideoFormatEvent) - of org.dvb.media.VideoFormatListener 608  
 register(Plugin, XletContext) - of org.dvb.application.plugins.XletSystemCall 1345  
 release() - of org.dvb.net.rc.ConnectionRCInterface 698  
 releaseDevice() - of org.havi.ui.HScreenDevice 1056  
 remove(EventListener) - of org.havi.ui.HEventMulticaster 871

remove(HActionListener, HActionListener) - of org.havi.ui.HEventMulticaster 871  
 remove(HAdjustmentListener, HAdjustmentListener) - of org.havi.ui.HEventMulticaster 871  
 remove(HBackgroundImageListener, HBackgroundImageListener) - of org.havi.ui.HEventMulticaster 872  
 remove(HFocusListener, HFocusListener) - of org.havi.ui.HEventMulticaster 872  
 remove(HItemListener, HItemListener) - of org.havi.ui.HEventMulticaster 872  
 remove(HKeyListener, HKeyListener) - of org.havi.ui.HEventMulticaster 872  
 remove(HScreenConfigurationListener, HScreenConfigurationListener) - of org.havi.ui.HEventMulticaster 872  
 remove(HScreenLocationModifiedListener, HScreenLocationModifiedListener) - of org.havi.ui.HEventMulticaster 873  
 remove(HTextListener, HTextListener) - of org.havi.ui.HEventMulticaster 873  
 remove(HToggleButton) - of org.havi.ui.HToggleGroup 1155  
 remove(ResourceStatusListener, ResourceStatusListener) - of org.havi.ui.HEventMulticaster 873  
 remove(String) - of org.dvb.user.Preference 483  
 remove(WindowListener, WindowListener) - of org.havi.ui.HEventMulticaster 873  
 removeAdjustmentListener(HAdjustmentListener) - of org.havi.ui.HAdjustmentValue 812  
 removeAdjustmentListener(HAdjustmentListener) - of org.havi.ui.HRangeValue 1016  
 removeAll() - of org.dvb.user.Preference 483  
 removeAllArrowKeys() - of org.dvb.event.UserEventRepository 467  
 removeAllColourKeys() - of org.dvb.event.UserEventRepository 468  
 removeAllItems() - of org.havi.ui.HListGroup 955  
 removeAllNumericKeys() - of org.dvb.event.UserEventRepository 468  
 removeAppStateChangeListener(AppStateChangeListener) - of org.dvb.application.AppProxy 722  
 removeBouquetMonitoringListener(SIMonitoringListener, int) - of org.dvb.si.SIDatabase 513  
 removeConnectionListener(ConnectionListener) - of org.dvb.net.rc.ConnectionRCInterface 698  
 removeEventPresentFollowingMonitoringListener(SIMonitoringListener, int, int, int) - of org.dvb.si.SIDatabase 513  
 removeEventScheduleMonitoringListener(SIMonitoringListener, int, int, int) - of org.dvb.si.SIDatabase 513  
 removeExclusiveAccessToAWTEvent(ResourceClient) - of org.dvb.event.EventManager 459  
 removeHActionListener(HActionListener) - of org.havi.ui.HActionable 802  
 removeHActionListener(HActionListener) - of org.havi.ui.HGraphicButton 900  
 removeHActionListener(HActionListener) - of org.havi.ui.HTextButton 1129  
 removeHActionListener(HActionListener) - of org.havi.ui.HToggleButton 1149  
 removeInnerApplicationListener(InnerApplicationListener) - of org.dvb.application.inner.InnerApplicationContainer 1323  
 removeInternal(EventListener, EventListener) - of org.havi.ui.HEventMulticaster 873  
 removeInternetClientListener(InternetClientListener) - of org.dvb.internet.InternetClient 1377  
 removeItem(int) - of org.havi.ui.HListGroup 955  
 removeItemListener(HItemListener) - of org.havi.ui.HItemValue 940  
 removeItemListener(HItemListener) - of org.havi.ui.HListGroup 956  
 removeKey(int) - of org.dvb.event.UserEventRepository 468  
 removeListener(AppsDatabaseEventListener) - of org.dvb.application.AppsDatabase 730  
 removeNetworkMonitoringListener(SIMonitoringListener, int) - of org.dvb.si.SIDatabase 514  
 removeNPTListener(NPTListener) - of org.dvb.dsmcc.DSMCCStream 640  
 removeObjectChangeEventListeners(ObjectChangeEventListeners) - of org.dvb.dsmcc.DSMCCObject 635  
 removeOnScreenLocationModifiedListener(HScreenLocationModifiedListener) - of org.havi.ui.HVideoComponent 1162  
 removePMTServiceMonitoringListener(SIMonitoringListener, int, int, int) - of org.dvb.si.SIDatabase 514  
 removeResourceStatusEventListener(ResourceStatusListener) - of org.dvb.event.EventManager 459  
 removeResourceStatusEventListener(ResourceStatusListener) - of org.dvb.net.rc.RCInterfaceManager 706  
 removeResourceStatusEventListener(ResourceStatusListener) - of org.havi.ui.HScreenDevice 1056  
 removeScreenConfigurationListener(HScreenConfigurationListener) - of org.havi.ui.HScreenDevice 1056  
 removeService() - of org.dvb.application.storage.StoredApplicationService 1362  
 removeServiceMonitoringListener(SIMonitoringListener, int, int) - of org.dvb.si.SIDatabase 514

removeShortcut(int) - of org.dvb.application.inner.DVBScene 1318  
 removeShortcut(int) - of org.havi.ui.HScene 1030  
 removeSubtitleListener(SubtitleListener) - of org.dvb.media.SubtitlingEventControl 601  
 removeTextOverflowListener(TextOverflowListener) - of org.dvb.ui.DVBTextLayoutManager 780  
 removeToggleGroup() - of org.havi.ui.HToggleButton 1149  
 removeUserEvent(UserEvent) - of org.dvb.event.UserEventRepository 468  
 removeUserEventListener(UserEventListener) - of org.dvb.event.EventManager 459  
 removeUserPreferenceChangeListener(UserPreferenceChangeListener) - of org.dvb.user.UserPreferenceManager 488  
 removeVideoFormatListener(VideoFormatListener) - of org.dvb.media.VideoFormatControl 606  
 removeWindowListener(WindowListener) - of org.dvb.application.inner.DVBScene 1319  
 removeWindowListener(WindowListener) - of org.havi.ui.HScene 1030  
 render(String, Graphics, HVisible, Insets) - of org.dvb.ui.DVBTextLayoutManager 780  
 render(String, Graphics, HVisible, Insets) - of org.havi.ui.HDefaultTextLayoutManager 860  
 render(String, Graphics, HVisible, Insets) - of org.havi.ui.HTextLayoutManager 1134  
 REPEAT\_INFINITE - of org.havi.ui.HAnimateEffect 815  
 report(HScreenConfigurationEvent) - of org.havi.ui.event.HScreenConfigurationListener 1239  
 report(HScreenConfigurationEvent) - of org.havi.ui.HEventMulticaster 874  
 report(HScreenLocationModifiedEvent) - of org.havi.ui.event.HScreenLocationModifiedListener 1244  
 report(HScreenLocationModifiedEvent) - of org.havi.ui.HEventMulticaster 874  
 RepositoryDescriptor - of org.dvb.event 461  
 REQUIRED - of org.havi.ui.HSceneTemplate 1038  
 REQUIRED - of org.havi.ui.HScreenConfigTemplate 1048  
 REQUIRED\_NOT - of org.havi.ui.HScreenConfigTemplate 1048  
 reserve(ResourceClient, Object) - of org.dvb.net.rc.ConnectionRCInterface 698  
 reserveDevice(ResourceClient) - of org.havi.ui.HScreenDevice 1057  
 RESIZE\_ARBITRARY - of org.havi.ui.HVisible 1183  
 RESIZE\_NONE - of org.havi.ui.HVisible 1183  
 RESIZE\_PRESERVE\_ASPECT - of org.havi.ui.HVisible 1183  
 resizeScene(HScene, HSceneTemplate) - of org.havi.ui.HSceneFactory 1036  
 resume() - of org.dvb.application.AppProxy 723  
 retrieveActualSINetwork(short, Object, SIRetrievalListener, short[]) - of org.dvb.si.SIDatabase 515  
 retrieveActualSIServices(short, Object, SIRetrievalListener, short[]) - of org.dvb.si.SIDatabase 515  
 retrieveActualSITransportStream(short, Object, SIRetrievalListener, short[]) - of org.dvb.si.SIDatabase 516  
 retrieveDescriptors(short, Object, SIRetrievalListener) - of org.dvb.si.SIBouquet 506  
 retrieveDescriptors(short, Object, SIRetrievalListener) - of org.dvb.si.SIInformation 533  
 retrieveDescriptors(short, Object, SIRetrievalListener) - of org.dvb.si.SINetwork 544  
 retrieveDescriptors(short, Object, SIRetrievalListener, short[]) - of org.dvb.si.SIBouquet 507  
 retrieveDescriptors(short, Object, SIRetrievalListener, short[]) - of org.dvb.si.SIInformation 533  
 retrieveDescriptors(short, Object, SIRetrievalListener, short[]) - of org.dvb.si.SINetwork 544  
 retrieveDetails(SIRequestor) - of org.dvb.application.storage.StoredApplicationService 1362  
 retrieveDetails(SIRequestor) - of org.dvb.internet.InternetClientService 1382  
 retrieveFollowingSIEvent(short, Object, SIRetrievalListener, short[]) - of org.dvb.si.SIService 557  
 retrievePMTElementaryStreams(short, Object, SIRetrievalListener, DvbLocator, short[]) - of org.dvb.si.SIDatabase 517  
 retrievePMTElementaryStreams(short, Object, SIRetrievalListener, int, int, short[]) - of org.dvb.si.SIDatabase 517  
 retrievePMTElementaryStreams(short, Object, SIRetrievalListener, short[]) - of org.dvb.si.PMTService 503  
 retrievePMTService(short, Object, SIRetrievalListener, DvbLocator, short[]) - of org.dvb.si.SIDatabase 518  
 retrievePMTService(short, Object, SIRetrievalListener, short[]) - of org.dvb.si.SIService 558  
 retrievePMTServices(short, Object, SIRetrievalListener, int, short[]) - of org.dvb.si.SIDatabase 519  
 retrievePresentSIEvent(short, Object, SIRetrievalListener, short[]) - of org.dvb.si.SIService 559  
 retrieveScheduledSIEvents(short, Object, SIRetrievalListener, short[], Date, Date) - of org.dvb.si.SIService 559  
 retrieveSIBouquets(short, Object, SIRetrievalListener, int, short[]) - of org.dvb.si.SIDatabase 520

retrieveSIBouquetTransportStreams(short, Object, SIRetrievalListener, short[]) - of org.dvb.si.SIBouquet 507  
 retrieveSINetworks(short, Object, SIRetrievalListener, int, short[]) - of org.dvb.si.SIDatabase 520  
 retrieveSIService(short, Object, SIRetrievalListener, DvbLocator, short[]) - of org.dvb.si.SIDatabase 521  
 retrieveSIService(short, Object, SIRetrievalListener, short[]) - of org.dvb.si.SIEvent 527  
 retrieveSIServices(short, Object, SIRetrievalListener, int, int, int, short[]) - of org.dvb.si.SIDatabase 522  
 retrieveSIServices(short, Object, SIRetrievalListener, short[]) - of org.dvb.si.SITransportStream 568  
 retrieveSITimeFromTDT(short, Object, SIRetrievalListener) - of org.dvb.si.SIDatabase 523  
 retrieveSITimeFromTOT(short, Object, SIRetrievalListener, short[]) - of org.dvb.si.SIDatabase 523  
 retrieveSITransportStreamDescription(short, Object, SIRetrievalListener, short[]) - of org.dvb.si.SIDatabase 524  
 retrieveSITransportStreams(short, Object, SIRetrievalListener, short[]) - of org.dvb.si.SINetwork 545  
 run(Document) - of org.dvb.dom.bootstrap.DocumentAction 1307  
 run(Document[]) - of org.dvb.dom.bootstrap.MultipleDocumentsAction 1310  
 RUNNING - of org.dvb.si.SIRunningStatus 554  
 SATELLITE\_DELIVERY\_SYSTEM - of org.dvb.si.DescriptorTag 498  
 SCENE\_PIXEL\_DIMENSION - of org.havi.ui.HSceneTemplate 1038  
 SCENE\_PIXEL\_LOCATION - of org.havi.ui.HSceneTemplate 1039  
 SCENE\_SCREEN\_DIMENSION - of org.havi.ui.HSceneTemplate 1039  
 SCENE\_SCREEN\_LOCATION - of org.havi.ui.HSceneTemplate 1039  
 SCHEDULED\_EVENT - of org.dvb.si.SIMonitoringType 542  
 SCREEN\_RECTANGLE - of org.havi.ui.HScreenConfigTemplate 1048  
 SCROLLBAR\_BEHAVIOR - of org.havi.ui.HStaticRange 1102  
 SECAM - of org.dvb.si.SIServiceType 562  
 selectGroup(URL) - of org.dvb.internet.UsenetClient 1389  
 selectionChanged(HItemEvent) - of org.havi.ui.event.HItemListener 1220  
 selectionChanged(HItemEvent) - of org.havi.ui.HEventMulticaster 874  
 selectMessage(URL) - of org.dvb.internet.UsenetClient 1389  
 ServerDeliveryErrorEvent - of org.dvb.dsmcc 669  
 ServerDeliveryErrorEvent(DSMCCObject) - of org.dvb.dsmcc.ServerDeliveryErrorEvent 669  
 ServerDeliveryException - of org.dvb.dsmcc 670  
 ServerDeliveryException() - of org.dvb.dsmcc.ServerDeliveryException 670  
 ServerDeliveryException(String) - of org.dvb.dsmcc.ServerDeliveryException 670  
 SERVICE - of org.dvb.si.DescriptorTag 498  
 SERVICE - of org.dvb.si.SIMonitoringType 542  
 SERVICE\_LIST - of org.dvb.si.DescriptorTag 498  
 SERVICE\_MOVE - of org.dvb.si.DescriptorTag 498  
 ServiceAlreadyExistsException - of org.dvb.application.storage 1359  
 ServiceAlreadyExistsException() - of org.dvb.application.storage.ServiceAlreadyExistsException 1359  
 ServiceAlreadyExistsException(String) - of org.dvb.application.storage.ServiceAlreadyExistsException 1359  
 ServiceDomain - of org.dvb.dsmcc 671  
 ServiceDomain() - of org.dvb.dsmcc.ServiceDomain 671  
 ServiceRemovedEvent - of org.dvb.media 592  
 ServiceRemovedEvent(Controller) - of org.dvb.media.ServiceRemovedEvent 592  
 ServiceRemovedEvent(Controller, MediaLocator) - of org.dvb.media.ServiceRemovedEvent 592  
 ServiceXFRErrorEvent - of org.dvb.dsmcc 675  
 ServiceXFRErrorEvent(DSMCCObject, ServiceXFRReference) - of org.dvb.dsmcc.ServiceXFRErrorEvent 675  
 ServiceXFRException - of org.dvb.dsmcc 676  
 ServiceXFRException(byte[], String) - of org.dvb.dsmcc.ServiceXFRException 676  
 ServiceXFRException(Locator, int, String) - of org.dvb.dsmcc.ServiceXFRException 676  
 ServiceXFRReference - of org.dvb.dsmcc 678  
 ServiceXFRReference(byte[], String) - of org.dvb.dsmcc.ServiceXFRReference 678  
 ServiceXFRReference(Locator, int, String) - of org.dvb.dsmcc.ServiceXFRReference 678  
 set(byte[]) - of org.havi.ui.HSound 1086  
 setActionCommand(String) - of org.havi.ui.HActionable 802

setActionCommand(String) - of org.havi.ui.HGraphicButton 900  
 setActionCommand(String) - of org.havi.ui.HTextButton 1129  
 setActionCommand(String) - of org.havi.ui.HToggleButton 1149  
 setActionSound(HSound) - of org.havi.ui.HActionable 803  
 setActionSound(HSound) - of org.havi.ui.HGraphicButton 901  
 setActionSound(HSound) - of org.havi.ui.HTextButton 1129  
 setActionSound(HSound) - of org.havi.ui.HToggleButton 1150  
 setAdjustmentSound(HSound) - of org.havi.ui.HAdjustmentValue 812  
 setAdjustmentSound(HSound) - of org.havi.ui.HRangeValue 1016  
 setAdjustMode(boolean) - of org.havi.ui.HAdjustmentInputPreferred 809  
 setAdjustMode(boolean) - of org.havi.ui.HRangeValue 1017  
 setAnimateContent(Image[], int) - of org.havi.ui.HVisible 1190  
 setBackgroundConfiguration(HBackgroundConfiguration) - of org.havi.ui.HBackgroundDevice 837  
 setBackgroundImage(Image) - of org.dvb.application.inner.DVBScene 1319  
 setBackgroundImage(Image) - of org.havi.ui.HScene 1031  
 setBackgroundMode(int) - of org.dvb.application.inner.DVBScene 1319  
 setBackgroundMode(int) - of org.havi.ui.HScene 1031  
 setBackgroundMode(int) - of org.havi.ui.HVisible 1190  
 setBehavior(int) - of org.havi.ui.HStaticRange 1104  
 setBlockIncrement(int) - of org.havi.ui.HAdjustmentValue 812  
 setBlockIncrement(int) - of org.havi.ui.HRangeValue 1017  
 setCaretCharPosition(int) - of org.havi.ui.HSinglelineEntry 1075  
 setClipRegion(Rectangle) - of org.dvb.media.VideoPresentationControl 612  
 setClipRegion(Rectangle) - of org.dvb.media.VideoTransformation 615  
 setCoherentScreenConfigurations(HScreenConfiguration[]) - of org.havi.ui.HScreen 1045  
 setColor(Color) - of org.dvb.ui.DVBGraphics 773  
 setColor(Color) - of org.havi.ui.event.HEventRepresentation 1211  
 setColor(Color) - of org.havi.ui.HBackgroundConfiguration 834  
 setColor(Color) - of org.havi.ui.HStillImageBackgroundConfiguration 1112  
 setContent(Object, int) - of org.havi.ui.HVisible 1190  
 setCurrent(HToggleButton) - of org.havi.ui.HToggleGroup 1155  
 setCurrentItem(int) - of org.havi.ui.HListGroup 956  
 setDefaultLook(HAnimateLook) - of org.havi.ui.HAnimation 827  
 setDefaultLook(HAnimateLook) - of org.havi.ui.HStaticAnimation 1094  
 setDefaultLook(HGraphicLook) - of org.havi.ui.HGraphicButton 901  
 setDefaultLook(HGraphicLook) - of org.havi.ui.HIcon 926  
 setDefaultLook(HGraphicLook) - of org.havi.ui.HStaticIcon 1099  
 setDefaultLook(HGraphicLook) - of org.havi.ui.HToggleButton 1150  
 setDefaultLook(HListGroupLook) - of org.havi.ui.HListGroup 956  
 setDefaultLook(HMultilineEntryLook) - of org.havi.ui.HMultilineEntry 980  
 setDefaultLook(HRangeLook) - of org.havi.ui.HRange 1003  
 setDefaultLook(HRangeLook) - of org.havi.ui.HRangeValue 1017  
 setDefaultLook(HRangeLook) - of org.havi.ui.HStaticRange 1104  
 setDefaultLook(HSinglelineEntryLook) - of org.havi.ui.HSinglelineEntry 1075  
 setDefaultLook(HTextLook) - of org.havi.ui.HStaticText 1109  
 setDefaultLook(HTextLook) - of org.havi.ui.HText 1121  
 setDefaultLook(HTextLook) - of org.havi.ui.HTextButton 1129  
 setDefaultSize(Dimension) - of org.havi.ui.HVisible 1190  
 setDelay(int) - of org.havi.ui.HAnimateEffect 816  
 setDelay(int) - of org.havi.ui.HFlatEffectMatte 879  
 setDelay(int) - of org.havi.ui.HImageEffectMatte 930  
 setDelay(int) - of org.havi.ui.HStaticAnimation 1095  
 setDVBComposite(DVBAlphaComposite) - of org.dvb.ui.DVBGraphics 773  
 setEchoChar(char) - of org.havi.ui.HSinglelineEntry 1075  
 setEditMode(boolean) - of org.havi.ui.HKeyboardInputPreferred 943



setEditMode(boolean) - of org.havi.ui.HSinglelineEntry 1075  
 setEnabled(boolean) - of org.havi.ui.HToggleGroup 1155  
 setExpirationDate(Date) - of org.dvb.io.persistent.FileAttributes 474  
 setFileAttributes(FileAttributes, File) - of org.dvb.io.persistent.FileAttributes 474  
 setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) - of org.dvb.application.inner.InnerApplicationContainer 1323  
 setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) - of org.havi.ui.HAnimation 827  
 setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) - of org.havi.ui.HGraphicButton 901  
 setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) - of org.havi.ui.HIcon 926  
 setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) - of org.havi.ui.HListGroup 957  
 setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) - of org.havi.ui.HNavigable 989  
 setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) - of org.havi.ui.HRange 1003  
 setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) - of org.havi.ui.HRangeValue 1017  
 setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) - of org.havi.ui.HSinglelineEntry 1075  
 setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) - of org.havi.ui.HText 1121  
 setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) - of org.havi.ui.HTextButton 1129  
 setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) - of org.havi.ui.HToggleButton 1150  
 setForcedSelection(boolean) - of org.havi.ui.HToggleGroup 1156  
 setGainFocusSound(HSound) - of org.dvb.application.inner.InnerApplicationContainer 1324  
 setGainFocusSound(HSound) - of org.havi.ui.HAnimation 828  
 setGainFocusSound(HSound) - of org.havi.ui.HGraphicButton 902  
 setGainFocusSound(HSound) - of org.havi.ui.HIcon 926  
 setGainFocusSound(HSound) - of org.havi.ui.HListGroup 957  
 setGainFocusSound(HSound) - of org.havi.ui.HNavigable 990  
 setGainFocusSound(HSound) - of org.havi.ui.HRange 1004  
 setGainFocusSound(HSound) - of org.havi.ui.HRangeValue 1018  
 setGainFocusSound(HSound) - of org.havi.ui.HSinglelineEntry 1076  
 setGainFocusSound(HSound) - of org.havi.ui.HText 1121  
 setGainFocusSound(HSound) - of org.havi.ui.HTextButton 1130  
 setGainFocusSound(HSound) - of org.havi.ui.HToggleButton 1151  
 setGraphicContent(Image, int) - of org.havi.ui.HVisible 1191  
 setGraphicsConfiguration(HEmulatedGraphicsConfiguration) - of org.havi.ui.HEmulatedGraphicsDevice 864  
 setGraphicsConfiguration(HGraphicsConfiguration) - of org.havi.ui.HGraphicsDevice 921  
 setHomepage(URL) - of org.dvb.internet.WWWBrowserService 1394  
 setHorizontalAlign(int) - of org.dvb.ui.DVBTextLayoutManager 780  
 setHorizontalAlignment(int) - of org.havi.ui.HVisible 1191  
 setHorizontalTabSpacing(int) - of org.dvb.ui.DVBTextLayoutManager 781  
 setIcon(Image) - of org.havi.ui.HListElement 945  
 setIconSize(Dimension) - of org.havi.ui.HListGroup 957  
 setInsets(Insets) - of org.dvb.ui.DVBTextLayoutManager 781  
 setInteractionState(int) - of org.havi.ui.HVisible 1191  
 setItemSelected(int, boolean) - of org.havi.ui.HListGroup 958  
 setLabel(String) - of org.havi.ui.HListElement 945  
 setLabelSize(Dimension) - of org.havi.ui.HListGroup 958  
 setLetterSpace(int) - of org.dvb.ui.DVBTextLayoutManager 781  
 setLineOrientation(int) - of org.dvb.ui.DVBTextLayoutManager 781  
 setLineSpace(int) - of org.dvb.ui.DVBTextLayoutManager 781  
 setListContent(HListElement[]) - of org.havi.ui.HListGroup 958  
 setLocation(float, float) - of org.havi.ui.HScreenPoint 1061  
 setLocation(float, float) - of org.havi.ui.HScreenRectangle 1063  
 setLook(HLook) - of org.havi.ui.HListGroup 958  
 setLook(HLook) - of org.havi.ui.HMultilineEntry 980  
 setLook(HLook) - of org.havi.ui.HSinglelineEntry 1076  
 setLook(HLook) - of org.havi.ui.HStaticAnimation 1095

setLook(HLook) - of org.havi.ui.HStaticIcon 1099  
setLook(HLook) - of org.havi.ui.HStaticRange 1104  
setLook(HLook) - of org.havi.ui.HStaticText 1109  
setLook(HLook) - of org.havi.ui.HVisible 1192  
setLookData(Object, Object) - of org.havi.ui.HVisible 1192  
setLoseFocusSound(HSound) - of org.dvb.application.inner.InnerApplicationContainer 1324  
setLoseFocusSound(HSound) - of org.havi.ui.HAnimation 828  
setLoseFocusSound(HSound) - of org.havi.ui.HGraphicButton 902  
setLoseFocusSound(HSound) - of org.havi.ui.HIcon 927  
setLoseFocusSound(HSound) - of org.havi.ui.HListGroup 959  
setLoseFocusSound(HSound) - of org.havi.ui.HNavigable 990  
setLoseFocusSound(HSound) - of org.havi.ui.HRange 1004  
setLoseFocusSound(HSound) - of org.havi.ui.HRangeValue 1018  
setLoseFocusSound(HSound) - of org.havi.ui.HSinglelineEntry 1077  
setLoseFocusSound(HSound) - of org.havi.ui.HText 1122  
setLoseFocusSound(HSound) - of org.havi.ui.HTextButton 1130  
setLoseFocusSound(HSound) - of org.havi.ui.HToggleButton 1151  
setMatte(HMatte) - of org.havi.ui.HComponent 845  
setMatte(HMatte) - of org.havi.ui.HContainer 857  
setMatte(HMatte) - of org.havi.ui.HMatteLayer 976  
setMatteData(float) - of org.havi.ui.HFlatMatte 883  
setMatteData(float[]) - of org.havi.ui.HFlatEffectMatte 879  
setMatteData(Image) - of org.havi.ui.HImageMatte 936  
setMatteData(Image[]) - of org.havi.ui.HImageEffectMatte 931  
setMaxCharsPerLine(int) - of org.havi.ui.HSinglelineEntry 1077  
setMostFavourite(String) - of org.dvb.user.Preference 483  
setMove(int, HNavigable) - of org.dvb.application.inner.InnerApplicationContainer 1324  
setMove(int, HNavigable) - of org.havi.ui.HAnimation 829  
setMove(int, HNavigable) - of org.havi.ui.HGraphicButton 902  
setMove(int, HNavigable) - of org.havi.ui.HIcon 927  
setMove(int, HNavigable) - of org.havi.ui.HListGroup 959  
setMove(int, HNavigable) - of org.havi.ui.HNavigable 990  
setMove(int, HNavigable) - of org.havi.ui.HRange 1004  
setMove(int, HNavigable) - of org.havi.ui.HRangeValue 1019  
setMove(int, HNavigable) - of org.havi.ui.HSinglelineEntry 1077  
setMove(int, HNavigable) - of org.havi.ui.HText 1122  
setMove(int, HNavigable) - of org.havi.ui.HTextButton 1131  
setMove(int, HNavigable) - of org.havi.ui.HToggleButton 1151  
setMultiSelection(boolean) - of org.havi.ui.HListGroup 959  
setOffset(Point) - of org.havi.ui.HImageMatte 936  
setOffset(Point, int) - of org.havi.ui.HImageEffectMatte 931  
setOrientation(int) - of org.havi.ui.HListGroup 959  
setOrientation(int) - of org.havi.ui.HOrientable 996  
setOrientation(int) - of org.havi.ui.HStaticRange 1104  
setPermissions(boolean, boolean, boolean, boolean, boolean, boolean) - of org.dvb.io.persistent.FileAccessPermissions 472  
setPermissions(FileAccessPermissions) - of org.dvb.io.persistent.FileAttributes 475  
setPlayMode(int) - of org.havi.ui.HAnimateEffect 816  
setPlayMode(int) - of org.havi.ui.HFlatEffectMatte 880  
setPlayMode(int) - of org.havi.ui.HImageEffectMatte 931  
setPlayMode(int) - of org.havi.ui.HStaticAnimation 1095  
setPosition(int) - of org.havi.ui.HAnimateEffect 816  
setPosition(int) - of org.havi.ui.HFlatEffectMatte 880  
setPosition(int) - of org.havi.ui.HImageEffectMatte 931  
setPosition(int) - of org.havi.ui.HStaticAnimation 1095

setPreference(int, int) - of org.havi.ui.HBackgroundConfigTemplate 832  
 setPreference(int, int) - of org.havi.ui.HGraphicsConfigTemplate 911  
 setPreference(int, int) - of org.havi.ui.HScreenConfigTemplate 1051  
 setPreference(int, Object, int) - of org.havi.ui.HGraphicsConfigTemplate 912  
 setPreference(int, Object, int) - of org.havi.ui.HSceneTemplate 1040  
 setPreference(int, Object, int) - of org.havi.ui.HScreenConfigTemplate 1051  
 setPreference(int, Object, int) - of org.havi.ui.HVideoConfigTemplate 1165  
 setPriority(int) - of org.dvb.io.persistent.FileAttributes 475  
 setRange(int, int) - of org.havi.ui.HStaticRange 1105  
 setReceiveBufferSize(DatagramSocket, int) - of org.dvb.net.DatagramSocketBufferControl 687  
 setRenderMode(int) - of org.dvb.application.inner.DVBScene 1319  
 setRenderMode(int) - of org.havi.ui.HScene 1031  
 setRepeatCount(int) - of org.havi.ui.HAnimateEffect 816  
 setRepeatCount(int) - of org.havi.ui.HFlatEffectMatte 880  
 setRepeatCount(int) - of org.havi.ui.HImageEffectMatte 932  
 setRepeatCount(int) - of org.havi.ui.HStaticAnimation 1096  
 setResizeMode(int) - of org.havi.ui.HVisible 1192  
 setRetrievalMode(int) - of org.dvb.dsmcc.DSMCCObject 635  
 setRGB(int, int, int) - of org.dvb.ui.DVBBufferedImage 765  
 setRGB(int, int, int, int, int[], int, int) - of org.dvb.ui.DVBBufferedImage 765  
 setScalingFactors(float, float) - of org.dvb.media.VideoTransformation 615  
 setScrollPosition(int) - of org.havi.ui.HListGroup 960  
 setSelectionMode(boolean) - of org.havi.ui.HListGroup 960  
 setSelectionMode(boolean) - of org.havi.ui.HSelectionInputPreferred 1065  
 setSelectionSound(HSound) - of org.havi.ui.HItemValue 940  
 setSelectionSound(HSound) - of org.havi.ui.HListGroup 960  
 setSize(float, float) - of org.havi.ui.HScreenDimension 1059  
 setSize(float, float) - of org.havi.ui.HScreenRectangle 1064  
 setStartCorner(int) - of org.dvb.ui.DVBTextLayoutManager 781  
 setString(String) - of org.havi.ui.event.HEventRepresentation 1211  
 setSwitchableState(boolean) - of org.havi.ui.HSwitchable 1115  
 setSwitchableState(boolean) - of org.havi.ui.HToggleButton 1152  
 setSymbol(Image) - of org.havi.ui.event.HEventRepresentation 1211  
 setTarget(ConnectionParameters) - of org.dvb.net.rc.ConnectionRCInterface 698  
 setTargetToDefault() - of org.dvb.net.rc.ConnectionRCInterface 698  
 setTextContent(String, int) - of org.havi.ui.HSinglelineEntry 1077  
 setTextContent(String, int) - of org.havi.ui.HVisible 1192  
 setTextLayoutManager(HTextLayoutManager) - of org.havi.ui.HVisible 1193  
 setTextWrapping(boolean) - of org.dvb.ui.DVBTextLayoutManager 781  
 setThumbOffsets(int, int) - of org.havi.ui.HStaticRange 1105  
 setToggleGroup(HToggleGroup) - of org.havi.ui.HToggleButton 1152  
 setType(int) - of org.havi.ui.event.HEventRepresentation 1212  
 setType(int) - of org.havi.ui.HImageHints 934  
 setType(int) - of org.havi.ui.HSinglelineEntry 1078  
 setUnitIncrement(int) - of org.havi.ui.HAdjustmentValue 813  
 setUnitIncrement(int) - of org.havi.ui.HRangeValue 1019  
 setUnsetActionSound(HSound) - of org.havi.ui.HSwitchable 1115  
 setUnsetActionSound(HSound) - of org.havi.ui.HToggleButton 1152  
 setValidInput(char[]) - of org.havi.ui.HSinglelineEntry 1078  
 setValue(int) - of org.havi.ui.HStaticRange 1105  
 setVerticalAlign(int) - of org.dvb.ui.DVBTextLayoutManager 782  
 setVerticalAlignment(int) - of org.havi.ui.HVisible 1193  
 setVideoConfiguration(HVideoConfiguration) - of org.havi.ui.HVideoDevice 1171  
 setVideoPosition(HScreenPoint) - of org.dvb.media.VideoTransformation 616  
 setVideoTransformation(VideoTransformation) - of org.dvb.media.BackgroundVideoPresentationControl

579

setVisible(boolean) - of org.havi.ui.HScene 1032  
 SHORT\_EVENT - of org.dvb.si.DescriptorTag 499  
 SHORT\_SMOOTHING\_BUFFER - of org.dvb.si.DescriptorTag 499  
 show() - of org.havi.ui.HScene 1032  
 showLook(Graphics, HVisible, int) - of org.havi.ui.HAnimateLook 821  
 showLook(Graphics, HVisible, int) - of org.havi.ui.HGraphicLook 907  
 showLook(Graphics, HVisible, int) - of org.havi.ui.HListGroupLook 965  
 showLook(Graphics, HVisible, int) - of org.havi.ui.HLook 971  
 showLook(Graphics, HVisible, int) - of org.havi.ui.HMultilineEntryLook 985  
 showLook(Graphics, HVisible, int) - of org.havi.ui.HRangeLook 1009  
 showLook(Graphics, HVisible, int) - of org.havi.ui.HSinglelineEntryLook 1082  
 showLook(Graphics, HVisible, int) - of org.havi.ui.HTextLook 1138  
 SIBouquet - of org.dvb.si 505  
 SIDatabase - of org.dvb.si 509  
 SIEvent - of org.dvb.si 525  
 SIException - of org.dvb.si 529  
 SIException() - of org.dvb.si.SIException 529  
 SIException(String) - of org.dvb.si.SIException 529  
 SIIllegalArgumentException - of org.dvb.si 530  
 SIIllegalArgumentException() - of org.dvb.si.SIIllegalArgumentException 530  
 SIIllegalArgumentException(String) - of org.dvb.si.SIIllegalArgumentException 530  
 SIInformation - of org.dvb.si 531  
 SIInvalidPeriodException - of org.dvb.si 535  
 SIInvalidPeriodException() - of org.dvb.si.SIInvalidPeriodException 535  
 SIInvalidPeriodException(String) - of org.dvb.si.SIInvalidPeriodException 535  
 SIIterator - of org.dvb.si 536  
 SILackOfResourcesEvent - of org.dvb.si 537  
 SILackOfResourcesEvent(Object, SIRequest) - of org.dvb.si.SILackOfResourcesEvent 537  
 SIMonitoringEvent - of org.dvb.si 538  
 SIMonitoringEvent(SIDatabase, byte, int, int, int, int, Date, Date) - of org.dvb.si.SIMonitoringEvent 538  
 SIMonitoringListener - of org.dvb.si 541  
 SIMonitoringType - of org.dvb.si 542  
 SINetwork - of org.dvb.si 543  
 SINotInCacheEvent - of org.dvb.si 547  
 SINotInCacheEvent(Object, SIRequest) - of org.dvb.si.SINotInCacheEvent 547  
 SIOBJECTNOTINTABLEEVENT - of org.dvb.si 548  
 SIOBJECTNOTINTABLEEVENT(Object, SIRequest) - of org.dvb.si.SIOBJECTNOTINTABLEEVENT 548  
 SIRequest - of org.dvb.si 549  
 SIRequestCancelledEvent - of org.dvb.si 550  
 SIRequestCancelledEvent(Object, SIRequest) - of org.dvb.si.SIRequestCancelledEvent 550  
 SIRetrievalEvent - of org.dvb.si 551  
 SIRetrievalEvent(Object, SIRequest) - of org.dvb.si.SIRetrievalEvent 551  
 SIRetrievalListener - of org.dvb.si 553  
 SIRunningStatus - of org.dvb.si 554  
 SIService - of org.dvb.si 555  
 SIServiceType - of org.dvb.si 561  
 SISuccessfulRetrieveEvent - of org.dvb.si 563  
 SISuccessfulRetrieveEvent(Object, SIRequest, SIIterator) - of org.dvb.si.SISuccessfulRetrieveEvent 563  
 SITableNotFoundEvent - of org.dvb.si 564  
 SITableNotFoundEvent(Object, SIRequest) - of org.dvb.si.SITableNotFoundEvent 564  
 SITableUpdatedEvent - of org.dvb.si 565  
 SITableUpdatedEvent(Object, SIRequest) - of org.dvb.si.SITableUpdatedEvent 565  
 SITime - of org.dvb.si 566  
 SITransportStream - of org.dvb.si 567

SITransportStreamBAT - of org.dvb.si 569  
 SITransportStreamDescription - of org.dvb.si 570  
 SITransportStreamNIT - of org.dvb.si 571  
 SIUtil - of org.dvb.si 572  
 size() - of org.dvb.application.AppsDatabase 730  
 SLIDER\_BEHAVIOR - of org.havi.ui.HStaticRange 1102  
 SMALL\_FORM\_VARIANTS - of org.havi.ui.HFontCapabilities 892  
 SmartCardPermission - of org.dvb.smartcard 1398  
 SmartCardPermission(String) - of org.dvb.smartcard.SmartCardPermission 1398  
 SmartCardPermission(String, String) - of org.dvb.smartcard.SmartCardPermission 1398  
 SPACING\_MODIFIER\_LETTERS - of org.havi.ui.HFontCapabilities 892  
 SPECIALS - of org.havi.ui.HFontCapabilities 892  
 SRC - of org.dvb.ui.DVBAlphaComposite 757  
 Src - of org.dvb.ui.DVBAlphaComposite 757  
 SRC\_IN - of org.dvb.ui.DVBAlphaComposite 757  
 SRC\_OUT - of org.dvb.ui.DVBAlphaComposite 757  
 SRC\_OVER - of org.dvb.ui.DVBAlphaComposite 758  
 SrcIn - of org.dvb.ui.DVBAlphaComposite 758  
 SrcOut - of org.dvb.ui.DVBAlphaComposite 758  
 SrcOver - of org.dvb.ui.DVBAlphaComposite 758  
 start() - of org.dvb.application.AppProxy 723  
 start() - of org.dvb.media.DripFeedDataSource 586  
 start() - of org.havi.ui.HAnimateEffect 817  
 start() - of org.havi.ui.HFlatEffectMatte 880  
 start() - of org.havi.ui.HImageEffectMatte 932  
 start() - of org.havi.ui.HStaticAnimation 1096  
 start(String[]) - of org.dvb.application.AppProxy 723  
 START\_CORNER\_LOWER\_LEFT - of org.dvb.ui.DVBTextLayoutManager 777  
 START\_CORNER\_LOWER\_RIGHT - of org.dvb.ui.DVBTextLayoutManager 777  
 START\_CORNER\_UPPER\_LEFT - of org.dvb.ui.DVBTextLayoutManager 777  
 START\_CORNER\_UPPER\_RIGHT - of org.dvb.ui.DVBTextLayoutManager 777  
 STARTED - of org.dvb.application.AppProxy 722  
 STARTS\_IN\_A\_FEW\_SECONDS - of org.dvb.si.SIRunningStatus 554  
 startTrigger(Date) - of org.dvb.application.DVBHTMLProxy 740  
 STATE\_CHANGE - of org.havi.ui.HVisible 1183  
 stateChange(AppStateChangeEvent) - of org.dvb.application.AppStateChangeListener 738  
 statusChanged(ResourceStatusEvent) - of org.havi.ui.HEventMulticaster 874  
 STILL\_IMAGE - of org.havi.ui.HBackgroundConfigTemplate 831  
 stop() - of org.dvb.media.DripFeedDataSource 586  
 stop() - of org.havi.ui.HAnimateEffect 817  
 stop() - of org.havi.ui.HFlatEffectMatte 881  
 stop() - of org.havi.ui.HImageEffectMatte 932  
 stop() - of org.havi.ui.HSound 1086  
 stop() - of org.havi.ui.HStaticAnimation 1096  
 stop(boolean) - of org.dvb.application.AppProxy 724  
 StopByResourceLossEvent - of org.dvb.media 594  
 StopByResourceLossEvent(Controller) - of org.dvb.media.StopByResourceLossEvent 594  
 StopByResourceLossEvent(Controller, MediaLocator) - of org.dvb.media.StopByResourceLossEvent 594  
 STORED - of org.dvb.application.storage.StoredApplicationService 1360  
 STORED\_APPLICATION\_SERVICE - of org.dvb.application.storage.StoredApplicationService 1360  
 StoredApplicationService - of org.dvb.application.storage 1360  
 StoredServiceCreator - of org.dvb.application.storage 1363  
 STREAM\_IDENTIFIER - of org.dvb.si.DescriptorTag 499  
 STREAM\_UNAVAILABLE - of org.dvb.media.PresentationChangedEvent 590  
 StreamEvent - of org.dvb.dsmcc 680

StreamEvent(DSMCCStreamEvent, long, String, int, byte[]) - of org.dvb.dsmcc.StreamEvent 680  
StreamEventListener - of org.dvb.dsmcc 682  
STUFFING - of org.dvb.si.DescriptorTag 499  
subscribe(String) - of org.dvb.internet.UsenetClientService 1391  
subscribe(String, StreamEventListener) - of org.dvb.dsmcc.DSMCCStreamEvent 642  
SubtitleAvailableEvent - of org.dvb.media 596  
SubtitleAvailableEvent(SubtitlingLanguageControl) - of org.dvb.media.SubtitleAvailableEvent 596  
SubtitleListener - of org.dvb.media 597  
SubtitleNotAvailableEvent - of org.dvb.media 598  
SubtitleNotAvailableEvent(SubtitlingLanguageControl) - of org.dvb.media.SubtitleNotAvailableEvent 598  
SubtitleNotSelectedEvent - of org.dvb.media 599  
SubtitleNotSelectedEvent(SubtitlingLanguageControl) - of org.dvb.media.SubtitleNotSelectedEvent 599  
SubtitleSelectedEvent - of org.dvb.media 600  
SubtitleSelectedEvent(SubtitlingLanguageControl) - of org.dvb.media.SubtitleSelectedEvent 600  
subtitleStatusChanged(EventObject) - of org.dvb.media.SubtitleListener 597  
SUBTITLING - of org.dvb.si.DescriptorTag 499  
SubtitlingEventControl - of org.dvb.media 601  
SuccessEvent - of org.dvb.dsmcc 683  
SuccessEvent(DSMCCObject) - of org.dvb.dsmcc.SuccessEvent 683  
SUPERSCRIPTS\_AND\_SUBSCRIPTS - of org.havi.ui.HFontCapabilities 892  
supportsArbitraryHorizontalScaling() - of org.dvb.media.VideoPresentationControl 612  
supportsArbitraryVerticalScaling() - of org.dvb.media.VideoPresentationControl 613  
supportsClipping() - of org.dvb.media.VideoPresentationControl 613  
synchronousLoad() - of org.dvb.dsmcc.DSMCCObject 636  
TAMIL - of org.havi.ui.HFontCapabilities 892  
TELEPHONE - of org.dvb.si.DescriptorTag 499  
TELETEXT - of org.dvb.si.DescriptorTag 499  
TELETEXT - of org.dvb.si.SIServiceType 562  
TELUGU - of org.havi.ui.HFontCapabilities 892  
terminate(String, int) - of org.dvb.test.DVBTest 1258  
terminatePlugin() - of org.dvb.application.plugins.Plugin 1335  
TERRESTRIAL\_DELIVERY\_SYSTEM - of org.dvb.si.DescriptorTag 499  
TestOpacity - of org.dvb.ui 787  
TEXT\_CARET\_CHANGE - of org.havi.ui.event.HTextEvent 1246  
TEXT\_CHANGE - of org.havi.ui.event.HTextEvent 1247  
TEXT\_CONTENT\_CHANGE - of org.havi.ui.HVisible 1183  
TEXT\_END\_CHANGE - of org.havi.ui.event.HTextEvent 1247  
TEXT\_FIRST - of org.havi.ui.event.HTextEvent 1247  
TEXT\_LAST - of org.havi.ui.event.HTextEvent 1247  
TEXT\_START\_CHANGE - of org.havi.ui.event.HTextEvent 1247  
TEXT\_VALUE\_CHANGE - of org.havi.ui.HVisible 1183  
textChanged(HTextEvent) - of org.havi.ui.event.HTextListener 1249  
textChanged(HTextEvent) - of org.havi.ui.HEventMulticaster 874  
TextOverflowListener - of org.dvb.ui 788  
TextualServiceIdentifierQuery - of org.dvb.si 573  
THAI - of org.havi.ui.HFontCapabilities 893  
THUMB\_OFFSETS\_CHANGE - of org.havi.ui.HVisible 1184  
TIME\_SHIFTED\_EVENT - of org.dvb.si.DescriptorTag 499  
TIME\_SHIFTED\_SERVICE - of org.dvb.si.DescriptorTag 499  
toString() - of org.dvb.application.AppID 720  
toString() - of org.dvb.ui.DVBBufferedImage 766  
toString() - of org.dvb.ui.DVBColor 770  
toString() - of org.dvb.ui.DVBGraphics 774  
toString() - of org.dvb.user.Preference 483  
trigger(Date, Object) - of org.dvb.application.DVBHTMLProxy 740

TunerPermission - of org.dvb.net.tuning 752  
TunerPermission(String) - of org.dvb.net.tuning.TunerPermission 752  
TunerPermission(String, String) - of org.dvb.net.tuning.TunerPermission 752  
TYPE\_ADVANCED - of org.dvb.ui.DVBBufferedImage 760  
TYPE\_BASE - of org.dvb.ui.DVBBufferedImage 760  
TYPE\_CATV - of org.dvb.net.rc.RCInterface 703  
TYPE\_DECT - of org.dvb.net.rc.RCInterface 703  
TYPE\_ISDN - of org.dvb.net.rc.RCInterface 703  
TYPE\_LMDS - of org.dvb.net.rc.RCInterface 703  
TYPE\_MATV - of org.dvb.net.rc.RCInterface 703  
TYPE\_PSTN - of org.dvb.net.rc.RCInterface 704  
UEF\_KEY\_EVENT - of org.dvb.event.UserEvent 462  
unbind(XletContext, String) - of org.dvb.io.ixc.IxcRegistry 1266  
UndeclaredThrowableException - of org.dvb.application.plugins 1341  
UndeclaredThrowableException(Throwable) - of org.dvb.application.plugins.UndeclaredThrowableException 1341  
UndeclaredThrowableException(Throwable, String) - of org.dvb.application.plugins.UndeclaredThrowableException 1341  
UNDEFINED - of org.dvb.si.SIRunningStatus 554  
ungroup() - of org.havi.ui.HContainer 858  
UNKNOWN - of org.dvb.si.SIServiceType 562  
UNKNOWN\_CHANGE - of org.havi.ui.HVisible 1184  
UnknownEventException - of org.dvb.dsmcc 684  
UnknownEventException() - of org.dvb.dsmcc.UnknownEventException 684  
UnknownEventException(String) - of org.dvb.dsmcc.UnknownEventException 684  
unload() - of org.dvb.dsmcc.DSMCCObject 636  
UNNECESSARY - of org.havi.ui.HSceneTemplate 1039  
unregister(Plugin, XletContext) - of org.dvb.application.plugins.XletSystemCall 1346  
UNRESOLVED - of org.dvb.test.DVBTest 1256  
unsubscribe(int, StreamEventListener) - of org.dvb.dsmcc.DSMCCStreamEvent 642  
unsubscribe(String, StreamEventListener) - of org.dvb.dsmcc.DSMCCStreamEvent 643  
UnsupportedDrawingOperationException - of org.dvb.ui 789  
UnsupportedDrawingOperationException(String) - of org.dvb.ui.UnsupportedDrawingOperationException 789  
UnsupportedPreferenceException - of org.dvb.user 484  
UnsupportedPreferenceException() - of org.dvb.user.UnsupportedPreferenceException 484  
UnsupportedPreferenceException(String) - of org.dvb.user.UnsupportedPreferenceException 484  
UNTESTED - of org.dvb.test.DVBTest 1256  
update(Graphics) - of org.havi.ui.HVisible 1193  
URLUnavailableEvent - of org.dvb.internet 1388  
URLUnavailableEvent(Object, URL) - of org.dvb.internet.URLUnavailableEvent 1388  
UsenetClient - of org.dvb.internet 1389  
UsenetClientService - of org.dvb.internet 1391  
UserEvent - of org.dvb.event 462  
UserEvent(Object, int, char) - of org.dvb.event.UserEvent 462  
UserEvent(Object, int, int, int, int) - of org.dvb.event.UserEvent 463  
UserEventListener - of org.dvb.event 465  
userEventReceived(UserEvent) - of org.dvb.event.UserEventListener 465  
UserEventRepository - of org.dvb.event 466  
UserEventRepository(String) - of org.dvb.event.UserEventRepository 466  
UserPreferenceChangeEvent - of org.dvb.user 485  
UserPreferenceChangeEvent(String) - of org.dvb.user.UserPreferenceChangeEvent 485  
UserPreferenceChangeListener - of org.dvb.user 486  
UserPreferenceManager - of org.dvb.user 487  
UserPreferencePermission - of org.dvb.user 489

UserPreferencePermission(String) - of org.dvb.user.UserPreferencePermission 489  
UserPreferencePermission(String, String) - of org.dvb.user.UserPreferencePermission 489  
UserRejectedInstallException - of org.dvb.application.storage 1364  
UserRejectedInstallException() - of org.dvb.application.storage.UserRejectedInstallException 1364  
UserRejectedInstallException(String) - of org.dvb.application.storage.UserRejectedInstallException 1364  
VALIGN\_BOTTOM - of org.havi.ui.HVisible 1184  
VALIGN\_CENTER - of org.havi.ui.HVisible 1184  
VALIGN\_JUSTIFY - of org.havi.ui.HVisible 1184  
VALIGN\_TOP - of org.havi.ui.HVisible 1184  
valueChanged(HAdjustmentEvent) - of org.havi.ui.event.HAdjustmentListener 1202  
valueChanged(HAdjustmentEvent) - of org.havi.ui.HEventMulticaster 875  
VERTICAL\_CENTER - of org.dvb.ui.DVBTextLayoutManager 777  
VERTICAL\_END\_ALIGN - of org.dvb.ui.DVBTextLayoutManager 777  
VERTICAL\_START\_ALIGN - of org.dvb.ui.DVBTextLayoutManager 777  
VIDEO\_GRAPHICS\_PIXEL\_ALIGNED - of org.havi.ui.HScreenConfigTemplate 1048  
VIDEO\_MIXING - of org.havi.ui.HGraphicsConfigTemplate 910  
VideoFormatControl - of org.dvb.media 602  
VideoFormatEvent - of org.dvb.media 607  
VideoFormatEvent(Object) - of org.dvb.media.VideoFormatEvent 607  
VideoFormatListener - of org.dvb.media 608  
VideoPresentationControl - of org.dvb.media 609  
VideoTransformation - of org.dvb.media 614  
VideoTransformation() - of org.dvb.media.VideoTransformation 614  
VideoTransformation(Rectangle, float, float, HScreenPoint) - of org.dvb.media.VideoTransformation 614  
VK\_BALANCE\_LEFT - of org.havi.ui.event.HRcEvent 1230  
VK\_BALANCE\_RIGHT - of org.havi.ui.event.HRcEvent 1230  
VK\_BASS\_BOOST\_DOWN - of org.havi.ui.event.HRcEvent 1230  
VK\_BASS\_BOOST\_UP - of org.havi.ui.event.HRcEvent 1230  
VK\_CHANNEL\_DOWN - of org.havi.ui.event.HRcEvent 1230  
VK\_CHANNEL\_UP - of org.havi.ui.event.HRcEvent 1230  
VK\_CLEAR\_FAVORITE\_0 - of org.havi.ui.event.HRcEvent 1230  
VK\_CLEAR\_FAVORITE\_1 - of org.havi.ui.event.HRcEvent 1230  
VK\_CLEAR\_FAVORITE\_2 - of org.havi.ui.event.HRcEvent 1231  
VK\_CLEAR\_FAVORITE\_3 - of org.havi.ui.event.HRcEvent 1231  
VK\_COLORED\_KEY\_0 - of org.havi.ui.event.HRcEvent 1231  
VK\_COLORED\_KEY\_1 - of org.havi.ui.event.HRcEvent 1231  
VK\_COLORED\_KEY\_2 - of org.havi.ui.event.HRcEvent 1231  
VK\_COLORED\_KEY\_3 - of org.havi.ui.event.HRcEvent 1231  
VK\_COLORED\_KEY\_4 - of org.havi.ui.event.HRcEvent 1232  
VK\_COLORED\_KEY\_5 - of org.havi.ui.event.HRcEvent 1232  
VK\_DIMMER - of org.havi.ui.event.HRcEvent 1232  
VK\_DISPLAY\_SWAP - of org.havi.ui.event.HRcEvent 1232  
VK\_EJECT\_TOGGLE - of org.havi.ui.event.HRcEvent 1232  
VK\_FADER\_FRONT - of org.havi.ui.event.HRcEvent 1232  
VK\_FADER\_REAR - of org.havi.ui.event.HRcEvent 1232  
VK\_FAST\_FWD - of org.havi.ui.event.HRcEvent 1232  
VK\_GO\_TO\_END - of org.havi.ui.event.HRcEvent 1232  
VK\_GO\_TO\_START - of org.havi.ui.event.HRcEvent 1233  
VK\_GUIDE - of org.havi.ui.event.HRcEvent 1233  
VK\_INFO - of org.havi.ui.event.HRcEvent 1233  
VK\_MUTE - of org.havi.ui.event.HRcEvent 1233  
VK\_PINP\_TOGGLE - of org.havi.ui.event.HRcEvent 1233  
VK\_PLAY - of org.havi.ui.event.HRcEvent 1233  
VK\_PLAY\_SPEED\_DOWN - of org.havi.ui.event.HRcEvent 1233  
VK\_PLAY\_SPEED\_RESET - of org.havi.ui.event.HRcEvent 1233



VK\_PLAY\_SPEED\_UP - of org.havi.ui.event.HRcEvent 1233  
VK\_POWER - of org.havi.ui.event.HRcEvent 1234  
VK\_RANDOM\_TOGGLE - of org.havi.ui.event.HRcEvent 1234  
VK\_RECALL\_FAVORITE\_0 - of org.havi.ui.event.HRcEvent 1234  
VK\_RECALL\_FAVORITE\_1 - of org.havi.ui.event.HRcEvent 1234  
VK\_RECALL\_FAVORITE\_2 - of org.havi.ui.event.HRcEvent 1234  
VK\_RECALL\_FAVORITE\_3 - of org.havi.ui.event.HRcEvent 1234  
VK\_RECORD - of org.havi.ui.event.HRcEvent 1234  
VK\_RECORD\_SPEED\_NEXT - of org.havi.ui.event.HRcEvent 1234  
VK\_REWIND - of org.havi.ui.event.HRcEvent 1234  
VK\_SCAN\_CHANNELS\_TOGGLE - of org.havi.ui.event.HRcEvent 1234  
VK\_SCREEN\_MODE\_NEXT - of org.havi.ui.event.HRcEvent 1235  
VK\_SPLIT\_SCREEN\_TOGGLE - of org.havi.ui.event.HRcEvent 1235  
VK\_STOP - of org.havi.ui.event.HRcEvent 1235  
VK\_STORE\_FAVORITE\_0 - of org.havi.ui.event.HRcEvent 1235  
VK\_STORE\_FAVORITE\_1 - of org.havi.ui.event.HRcEvent 1235  
VK\_STORE\_FAVORITE\_2 - of org.havi.ui.event.HRcEvent 1235  
VK\_STORE\_FAVORITE\_3 - of org.havi.ui.event.HRcEvent 1235  
VK\_SUBTITLE - of org.havi.ui.event.HRcEvent 1235  
VK\_SURROUND\_MODE\_NEXT - of org.havi.ui.event.HRcEvent 1235  
VK\_TELETEXT - of org.havi.ui.event.HRcEvent 1236  
VK\_TRACK\_NEXT - of org.havi.ui.event.HRcEvent 1236  
VK\_TRACK\_PREV - of org.havi.ui.event.HRcEvent 1236  
VK\_VIDEO\_MODE\_NEXT - of org.havi.ui.event.HRcEvent 1236  
VK\_VOLUME\_DOWN - of org.havi.ui.event.HRcEvent 1236  
VK\_VOLUME\_UP - of org.havi.ui.event.HRcEvent 1236  
VK\_WINK - of org.havi.ui.event.HRcEvent 1236  
widgetChanged(HVisible, HChangeData[]) - of org.havi.ui.HAnimateLook 822  
widgetChanged(HVisible, HChangeData[]) - of org.havi.ui.HGraphicLook 908  
widgetChanged(HVisible, HChangeData[]) - of org.havi.ui.HListGroupLook 966  
widgetChanged(HVisible, HChangeData[]) - of org.havi.ui.HLook 972  
widgetChanged(HVisible, HChangeData[]) - of org.havi.ui.HMultilineEntryLook 986  
widgetChanged(HVisible, HChangeData[]) - of org.havi.ui.HRangeLook 1010  
widgetChanged(HVisible, HChangeData[]) - of org.havi.ui.HSinglelineEntryLook 1083  
widgetChanged(HVisible, HChangeData[]) - of org.havi.ui.HTextLook 1139  
width - of org.havi.ui.HScreenDimension 1059  
width - of org.havi.ui.HScreenRectangle 1063  
windowActivated(WindowEvent) - of org.havi.ui.HEventMulticaster 875  
windowClosed(WindowEvent) - of org.havi.ui.HEventMulticaster 875  
windowClosing(WindowEvent) - of org.havi.ui.HEventMulticaster 875  
windowDeactivated(WindowEvent) - of org.havi.ui.HEventMulticaster 876  
windowDeiconified(WindowEvent) - of org.havi.ui.HEventMulticaster 876  
windowIconified(WindowEvent) - of org.havi.ui.HEventMulticaster 876  
windowOpened(WindowEvent) - of org.havi.ui.HEventMulticaster 876  
write(Preference) - of org.dvb.user.UserPreferenceManager 488  
WWW\_CLIENT - of org.dvb.internet.InternetServiceFilter 1384  
WWWBrowser - of org.dvb.internet 1392  
WWWBrowserService - of org.dvb.internet 1393  
x - of org.havi.ui.HScreenPoint 1061  
x - of org.havi.ui.HScreenRectangle 1063  
XletContainer - of org.dvb.application.plugins 1343  
XletContainer(Container) - of org.dvb.application.plugins.XletContainer 1343  
XletSystemCall - of org.dvb.application.plugins 1345  
XletSystemCall() - of org.dvb.application.plugins.XletSystemCall 1345  
y - of org.havi.ui.HScreenPoint 1061

y - of org.havi.ui.HScreenRectangle 1063

ZERO\_BACKGROUND\_IMPACT - of org.havi.ui.HScreenConfigTemplate 1049

ZERO\_GRAPHICS\_IMPACT - of org.havi.ui.HScreenConfigTemplate 1049

ZERO\_VIDEO\_IMPACT - of org.havi.ui.HScreenConfigTemplate 1049

# History

Document history		
V1.1.1	November 2001	Published