

ETSI TS 101 812 V1.2.1 (2002-06)

Technical Specification

Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.0.2

European Broadcasting Union



Union Européenne de Radio-Télévision

EBU·UER
DVB
Digital Video
Broadcasting



Reference

RTS/JTC-DVB-130

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 enregistrée à la
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 2002.

© European Broadcasting Union 2002.

All rights reserved.

DECT™, **PLUGTESTS™** and **UMTS™** are Trade Marks of ETSI registered for the benefit of its Members.
TIPHON™ and the **TIPHON logo** are Trade Marks currently being registered by ETSI for the benefit of its Members.
3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.

Contents

Intellectual Property Rights	26
Foreword	26
0 Introduction	26
0.1 Purpose	26
0.2 Application Areas	27
0.3 Profiles	27
1 Scope	28
2 References	28
3 Definitions and abbreviations	34
3.1 Definitions	34
3.2 Abbreviations	37
4 Conventions	38
5 Basic Architecture	39
5.1 Context	39
5.2 Architecture	40
5.2.1 Resources	40
5.2.2 System software	40
5.2.2.1 Application Manager	40
5.2.3 Application	40
5.3 Interfaces Between an MHP Application and the MHP System	42
5.4 Plug-ins	43
5.4.1 Security Model	44
6 Transport Protocols	45
6.1 Introduction	45
6.2 Broadcast Channel Protocols	45
6.2.1 MPEG-2 Transport Stream	46
6.2.2 MPEG-2 Sections	46
6.2.3 DSM-CC Private Data	46
6.2.4 DSM-CC Data Carousel	46
6.2.5 DSM-CC User-to-User Object Carousel	46
6.2.5.1 DVB-J class files	46
6.2.5.2 DVB-HTML document files	47
6.2.5.3 Loss of Carousel Behaviour	47
6.2.6 DVB Multiprotocol Encapsulation	47
6.2.7 Internet Protocol (IP)	47
6.2.8 User Datagram Protocol (UDP)	47
6.2.9 DVB Service Information	47
6.3 Interaction Channel Protocols	48
6.3.1 Network Dependent Protocols	48
6.3.2 Internet Protocol (IP)	48
6.3.3 Transmission Control Protocol (TCP)	48
6.3.4 UNO-RPC	48
6.3.5 UNO-CDR	48
6.3.6 DCM-CC User to User	48
6.3.7 Hypertext Transfer Protocol (HTTP)	49
6.3.7.1 HTTP 1.1	49
6.3.8 Service Specific	49
6.3.9 User Datagram Protocol (UDP)	49
7 Content formats	50
7.1 Static formats	50
7.1.1 Bitmap image formats	50

7.1.1.1	Image encoding restrictions	50
7.1.1.2	JPEG	50
7.1.1.3	PNG	50
7.1.1.4	GIF	50
7.1.2	MPEG-2 I-Frames	50
7.1.3	MPEG-2 Video "drips"	50
7.1.4	Monomedia format for audio clips	52
7.1.5	Monomedia format for text	52
7.1.5.1	Built-in character set	52
7.2	Broadcast streaming formats	52
7.2.1	Audio	52
7.2.2	Video	52
7.2.3	Subtitles	52
7.2.3.1	DVB Subtitles	52
7.2.3.2	Teletext	53
7.3	Resident fonts	53
7.4	Downloadable Fonts	53
7.5	Colour Representation	53
7.5.1	Background (informative)	53
7.5.2	Specification	54
7.5.2.1	The sRGB Reference Viewing Environment	54
7.5.2.2	Colourimetric Definitions and Encodings	54
7.6	MIME Types	56
7.6.1	Rationale	56
8	DVB-HTML	57
8.1	Status of DVB HTML	57
9	Application model	58
9.1	Broadcast MHP applications	58
9.1.1	Basic lifecycle control	58
9.1.2	Starting applications	58
9.1.3	Support for execution of multiple simultaneous applications	59
9.1.4	Stopping applications	59
9.1.4.1	A new service being selected replacing a previously selected one	59
9.1.4.2	The stopping of an application by another application	59
9.1.4.3	Changes in the application signalling to request a particular application be stopped	59
9.1.4.4	Stopping by the MHP terminal due to a shortage of resources	59
9.1.5	Persistence of Applications Across Service Boundaries	59
9.1.6	Management of autostarting	60
9.1.7	When tuning is not service selection!	61
9.1.8	DVB-J Applications and Service Selection	61
9.2	DVB-J Model	61
9.2.1	Starting DVB-J Applications	61
9.2.2	Stopping a DVB-J Application	61
9.2.3	DVB-J Application Lifecycle	62
9.2.3.1	Introduction	62
9.2.3.2	DVB-J Application Lifecycle State Machine	62
9.2.4	Xlet API	65
9.2.4.1	Xlet State Change Semantics	66
9.2.4.2	Xlet state change requests	66
9.2.5	Multiple application environment support	66
9.2.5.1	Control of DVB-J applications by other DVB-J applications	66
9.2.5.2	Input Focus management	66
9.2.5.3	Other resources management	67
9.2.5.4	VM implementation	67
9.3	DVB-HTML Model	67
9.3.1	The DVB-HTML Application	67
9.3.1.1	DVB-HTML Application	67
9.3.1.2	User agent	67

9.3.1.3	DVB-HTML Actor	67
9.3.1.4	Application boundary	68
9.3.1.4.1	Regular Expression Syntax	68
9.3.2	DVB-HTML Application Lifecycle	69
9.3.2.1	Introduction	69
9.3.2.2	Signalling	69
9.3.2.3	Lifecycle control	70
9.3.2.3.1	State diagram	70
9.3.3	The State Model	70
9.3.3.1	Loading	71
9.3.3.1.1	Name	71
9.3.3.1.2	Entry actions	71
9.3.3.1.3	Activities	71
9.3.3.1.4	Resources	71
9.3.3.1.5	Transitions	71
9.3.3.1.6	Comment	71
9.3.3.2	Active	71
9.3.3.2.1	Name	71
9.3.3.2.2	Activities	71
9.3.3.2.3	Entry actions	71
9.3.3.2.4	Resources	71
9.3.3.2.5	Transitions	72
9.3.3.2.6	Comment	72
9.3.3.3	Paused	72
9.3.3.3.1	Name	72
9.3.3.3.2	Activities	72
9.3.3.3.3	Resources	72
9.3.3.3.4	Transitions	72
9.3.3.3.5	Comment	72
9.3.3.4	Destroyed	73
9.3.3.4.1	Name	73
9.3.3.4.2	Activities	73
9.3.3.4.3	Resources	73
9.3.3.4.4	Transitions:	73
9.3.3.4.5	Comment	73
9.3.3.5	Killed	73
9.3.3.5.1	Name	73
9.3.3.5.2	Entry actions	73
9.3.3.5.3	Activities	73
9.3.3.5.4	Resources	73
9.3.3.5.5	Transitions	73
9.3.3.5.6	Comment	73
9.4	Inter application resource management	74
10	Application Signalling	75
10.1	Introduction	75
10.1.1	Summary of common signalling	75
10.1.2	Summary of additional signalling for DVB-J applications	75
10.1.3	Summary of additional signalling for DVB-HTML applications	75
10.1.4	Summary of additional signalling for applications carried via OC	75
10.1.5	Summary of additional signalling for applications carried via IP	76
10.1.6	How to add a new scheme (informative)	76
10.1.7	Service information	76
10.2	Program Specific Information	76
10.2.1	Application signalling stream	76
10.2.2	Data broadcast streams	76
10.3	Notation	77
10.3.1	reserved	77
10.3.2	reserved_future_use	77
10.4	Application Information Table	77

10.4.1	Data errors	77
10.4.2	AIT transmission and monitoring	77
10.4.3	Optimised AIT signalling	78
10.4.4	Visibility of AIT	78
10.4.5	Definition of sub-table for the AIT	78
10.4.6	Syntax of the AIT	78
10.4.7	Use of private descriptors in the AIT	80
10.4.8	Text encoding in AIT	80
10.5	Application identification	80
10.5.1	Encoding	80
10.5.2	Effects on life cycle	81
10.5.3	Authentication of application identification	81
10.6	Control of application life cycle	81
10.6.1	Entering and leaving the domain of an application	82
10.6.2	Dynamic control of the application life cycle	82
10.6.2.1	DVB-J	82
10.6.2.2	DVB-HTML	83
10.7	Generic descriptors	83
10.7.1	Application Signalling Descriptor	83
10.7.2	Data broadcast id descriptor	84
10.7.2.1	Generic descriptor	84
10.7.2.2	MHP data broadcast id descriptor	85
10.7.3	Application descriptor	85
10.7.4	User information descriptors	87
10.7.4.1	Application name descriptor	87
10.7.4.2	Application icons descriptor	88
10.7.5	External application authorisation descriptor	89
10.8	Transport protocol descriptors	90
10.8.1	Transport protocol descriptor	90
10.8.1.1	Transport via OC	91
10.8.1.2	Transport via IP	91
10.8.2	IP Routing Descriptors	92
10.8.2.1	Routing Descriptor IPv4	92
10.8.2.2	Routing Descriptor IPv6	93
10.8.3	Pre-fetch signalling	93
10.8.3.1	Introduction	93
10.8.3.2	Pre-fetch descriptor	94
10.8.3.3	DII location descriptor	94
10.9	DVB-J specific descriptors	95
10.9.1	DVB-J application descriptor	95
10.9.2	DVB-J application location descriptor	96
10.10	DVB-HTML Specific descriptors	97
10.10.1	DVB-HTML application descriptor	97
10.10.2	DVB-HTML application location descriptor	97
10.10.2.1	Example	98
10.10.2.2	Application Entry Point	98
10.10.3	DVB-HTML application boundary descriptor	99
10.11	Constant values	100
10.12	Service Information	101
10.12.1	Service identifier descriptor	101
11	DVB-J Platform	102
11.1	The Virtual Machine	102
11.2	General issues	102
11.2.1	Basic Considerations	102
11.2.2	Approach to Subsetting	103
11.2.3	Class Loading	103
11.2.4	Unloading	103
11.2.5	Event listeners	103
11.2.6	Event model in DAVIC APIs	103

11.2.7	Event model in DAVIC & DVB APIs	103
11.2.8	Tuning as a side-effect.	103
11.2.9	Intra application media resource management	104
11.2.10	Application thread priority	104
11.2.11	Text Encodings	104
11.3	Fundamental DVB-J APIs.	104
11.3.1	Java platform APIs	104
11.3.1.1	java.lang package.	104
11.3.1.2	java.lang.reflect package	106
11.3.1.3	java.util	106
11.3.1.4	java.util.zip.	106
11.3.1.5	java.io.	106
11.3.1.6	java.net.	106
11.3.1.7	java.beans.	108
11.3.1.8	java.math	108
11.3.2	MHP platform APIs.	108
11.3.2.1	org.dvb.lang	108
11.3.2.2	org.dvb.event	108
11.4	Presentation APIs	109
11.4.1	Graphical User Interface API	109
11.4.1.1	The Core GUI API.	109
11.4.1.2	TV user interface	110
11.4.1.3	Extended graphics	111
11.4.1.4	Handling of input events	111
11.4.2	Streamed Media API	113
11.4.2.1	Framework of solution.	113
11.4.2.2	Clarifications	113
11.4.2.3	Default media player behaviour.	113
11.4.2.4	Required controls for video drips	114
11.4.2.5	Extensions to the Framework	114
11.4.2.5.1	DVB specified extensions	114
11.4.2.5.2	Extensions in org.davic	114
11.4.2.5.3	Extensions in javax.tv	114
11.4.2.5.4	Required controls for broadcast profiles	115
11.4.2.5.5	Clarifications	116
11.4.2.6	Restrictions on the Framework for Broadcast	116
11.4.2.7	Intersection Between MediaSelectControl and SubtitlingLanguageControl / AudioLanguageControl.	117
11.4.2.8	Intersection between Streamed Media API and TV User Interface API	117
11.4.2.8.1	Basic Principles	117
11.4.2.8.2	TV Behaviour Control	118
11.4.2.8.3	Application Behaviour Control	118
11.4.2.8.4	Dynamic Behaviour	118
11.4.2.8.5	Resource Management Details.	118
11.5	Data Access APIs	118
11.5.1	Broadcast Transport Protocol Access API	118
11.5.1.1	Constraints on the java.io.File methods for broadcast carousels.	119
11.5.1.2	Methods dealing with write access	119
11.5.1.3	Behaviour following loss of a broadcast carousel	120
11.5.2	Support for Multicast IP over the Broadcast Channel.	120
11.5.3	Support for IP over the Return Channel	121
11.5.4	MPEG-2 Section Filter API.	121
11.5.5	Mid-Level Communications API	121
11.5.6	Persistent Storage API.	122
11.6	Service Information and Selection APIs	123
11.6.1	DVB Service Information API	123
11.6.2	Service Selection API	123
11.6.3	Tuning API	124

11.6.4	Conditional Access API	125
11.6.5	Protocol Independent SI API	125
11.7	Common Infrastructure APIs	126
11.7.1	APIs to support DVB-J application lifecycle	126
11.7.1.1	Xlet properties	126
11.7.1.2	Actions for DVB-J applications to perform in their destroy method.	126
11.7.2	Application discovery and launching APIs	127
11.7.3	Inter-Application communication API	128
11.7.3.1	Remote Call Semantics	128
11.7.3.1.1	Objects Passed by Remote Reference	128
11.7.3.1.2	Objects Passed by Remote Copy	129
11.7.3.1.3	Classloading Considerations	130
11.7.3.1.4	Thread Usage	130
11.7.3.1.5	Garbage Collection of Remote Objects	130
11.7.4	Basic MPEG Concepts	130
11.7.5	Resource Notification	131
11.7.6	Content Referencing	131
11.7.7	Common Error Reporting	132
11.8	Security	132
11.8.1	Basic Security	132
11.8.1.1	java.security	132
11.8.1.2	java.security.cert	133
11.8.1.3	Other classes	133
11.8.2	APIs to Support TLS / SSL Over the Return Channel	133
11.8.3	Additional permissions classes	133
11.8.4	General security issues	134
11.9	Other APIs	134
11.9.1	Timer Support	134
11.9.2	User Settings and Preferences API	134
11.9.3	Profile and version properties	134
11.9.3.1	Information on options	135
11.10	Java permissions	135
11.10.1	Permissions for unsigned applications	135
11.10.1.1	java.awt.AWTPermission	136
11.10.1.2	java.net.SocketPermission:	136
11.10.1.3	java.util.PropertyPermission	136
11.10.1.4	java.lang.RuntimePermission	136
11.10.1.5	java.io.SerializablePermission	136
11.10.1.6	java.io.FilePermission	136
11.10.1.7	javax.tv.media.MediaSelectPermission	136
11.10.1.8	javax.tv.service.ReadPermission	136
11.10.1.9	javax.tv.service.selection.ServiceContextPermission	136
11.10.1.10	java.util.Locale.setDefault	136
11.10.2	Additional Permissions for signed applications	136
11.10.2.1	java.util.PropertyPermission	136
11.10.2.2	java.io.FilePermission	137
11.10.2.3	org.dvb.net.ca.CAPermission	137
11.10.2.4	org.dvb.application.AppsControlPermission	137
11.10.2.5	org.dvb.net.rc.RCPermission	137
11.10.2.6	org.dvb.net.tuning.TunerPermission	137
11.10.2.7	javax.tv.service.selection.SelectPermission	137
11.10.2.8	org.dvb.user.UserPreferencePermission	138
11.10.2.9	java.net.SocketPermission	138
11.10.2.10	org.dvb.media.DripFeedPermission	138
11.11	Content referencing	138
11.11.1	Transport stream	139
11.11.2	Network	139
11.11.3	Bouquet	139

11.11.4	Service	140
11.11.4.1	MPEG/DVB specific service	140
11.11.4.2	Generic Service	141
11.11.5	DVB Event	141
11.11.6	MPEG elementary stream	142
11.11.7	File	142
11.11.8	Directory	143
11.11.9	Drip feed decoder	143
11.11.10	Irrelevant	143
11.11.11	Methods working on many Locator types	143
11.11.12	Support for the HTTP protocol in DVB-J	144
12	Security	145
12.1	Introduction	145
12.1.1	Overview of the security framework for applications	145
12.1.2	Overview of return channel security	145
12.2	Authentication of applications	145
12.2.1	Overview of authentication messages	145
12.2.1.1	Hash codes	146
12.2.1.2	Signatures	146
12.2.1.3	Certificates	146
12.2.1.4	Authentication of hierarchical file systems	147
12.3	Message transport	147
12.4	Detail of application authentication messages	148
12.4.1	HashFile	148
12.4.1.1	Description	148
12.4.1.2	HashFile location and naming conventions	149
12.4.1.3	Digest value computation rules	149
12.4.1.3.1	Example	150
12.4.1.4	Warning concerning grouping of objects under a single digest (Informative)	150
12.4.1.5	Special authentication rules	150
12.4.2	SignatureFile	151
12.4.2.1	Description	151
12.4.2.2	SignatureFile location and naming conventions	151
12.4.2.3	Supported algorithms	151
12.4.2.4	Signature computation rules	152
12.4.2.5	Authentication rules	152
12.4.3	CertificateFile	152
12.4.3.1	Description	152
12.4.3.2	ASN.1 encoding	152
12.4.3.3	Supported algorithms	152
12.4.3.4	Name matching	152
12.4.3.5	CertificateFile location and naming conventions	153
12.4.3.6	Authentication rules	153
12.4.4	Integration	153
12.5	Profile of X.509 certificates for authentication of applications	154
12.5.1	signatureAlgorithm	154
12.5.1.1	MD5 with RSA	154
12.5.1.2	SHA-1 with RSA	154
12.5.1.3	parameters	154
12.5.2	signatureValue	154
12.5.3	version	154
12.5.4	issuer	154
12.5.4.1	minimum requirement	154
12.5.4.2	certificate authority responsibility	154
12.5.5	validity	155
12.5.6	subject	155
12.5.7	SubjectPublic Key Info	155
12.5.7.1	rsaEncryption	155
12.5.7.2	subjectPublicKey	155

12.5.8	Unique Identifiers	155
12.5.9	Extensions	156
12.6	Security policy for applications	157
12.6.1	General principles	157
12.6.2	Permission request file	158
12.6.2.1	File encoding	158
12.6.2.1.1	Number representation	159
12.6.2.2	File integrity	159
12.6.2.3	Example	160
12.6.2.4	Permission request file name and location	160
12.6.2.5	Permission request file	161
12.6.2.5.1	Minimum permissions	161
12.6.2.5.2	Syntax and semantics	161
12.6.2.6	Credentials	161
12.6.2.7	File Access	163
12.6.2.7.1	Unsigned applications	163
12.6.2.7.2	Policy for signed applications	163
12.6.2.7.3	Permission request syntax	164
12.6.2.8	CA API	164
12.6.2.8.1	Unsigned applications	164
12.6.2.8.2	Signed applications	164
12.6.2.8.3	Conditional Access Permission syntax	165
12.6.2.9	Application lifecycle control policy	165
12.6.2.9.1	Unsigned applications	165
12.6.2.9.2	Default policy for Signed applications	165
12.6.2.9.3	Syntax	165
12.6.2.10	Return channel access policy	165
12.6.2.10.1	Unsigned applications	165
12.6.2.10.2	Signed applications	165
12.6.2.10.3	Return channel permission syntax	166
12.6.2.11	Tuning access policy	166
12.6.2.11.1	Unsigned applications	166
12.6.2.11.2	Signed applications	166
12.6.2.11.3	Tuner Permission syntax	166
12.6.2.12	Service selection policy	166
12.6.2.12.1	Unsigned applications	166
12.6.2.12.2	Signed applications	166
12.6.2.12.3	Service Selection Permission	166
12.6.2.13	Media API access policy	167
12.6.2.14	Inter-application communication policy	167
12.6.2.14.1	Unsigned applications	167
12.6.2.14.2	Signed applications	167
12.6.2.15	User Setting and Preferences access policy	167
12.6.2.15.1	Unsigned applications	167
12.6.2.15.2	Signed applications	167
12.6.2.15.3	Permission syntax	167
12.6.2.16	Network permissions	167
12.6.2.16.1	Unsigned applications	167
12.6.2.16.2	Signed applications	167
12.6.2.16.3	Permission syntax	168
12.6.2.17	Dripfeed permissions	168
12.6.2.17.1	Unsigned applications	168
12.6.2.17.2	Default policy for signed applications	168
12.6.2.17.3	Permission request syntax	168
12.7	Example of creating an application that can be authenticated	168
12.7.1	Scenario Example	168
12.7.2	Hashes and signature computations:	169
12.7.2.1	Computation of the hashes of the root/Xlet1/classes/subclasses directory	169
12.7.2.2	Computation of the hashes of the of root/Xlet1/classes directory	169

12.7.2.3	Computation of the hashes of the of root/Xlet1 directory	170
12.7.2.4	Computation of the signature.	170
12.8	MHP certification procedures	171
12.9	Certificate management.	171
12.9.1	Certificate Revocation Lists	171
12.9.1.1	Introduction (informative)	171
12.9.1.2	Distribution of CRLs (informative)	171
12.9.1.2.1	Distribution via return channel.	171
12.9.1.2.2	Distribution via MPEG stream.	171
12.9.1.3	CRL retention	172
12.9.1.3.1	Requirement	172
12.9.1.3.2	Storage requirement	172
12.9.1.3.3	Storage management	172
12.9.1.4	CRL file location and naming convention.	172
12.9.1.5	Operational model	172
12.9.1.6	Examples	173
12.9.1.6.1	Revocation of a broadcaster's certificate	173
12.9.1.6.2	Revocation of a CA's certificate.	173
12.9.1.7	CRL format	173
12.9.1.8	Profile of CRL	174
12.9.1.9	CRL Processing	174
12.9.2	Root certificate management.	175
12.9.2.1	Introduction	175
12.9.2.2	Security of RCMM	175
12.9.2.3	Format of RCMM	175
12.9.2.4	Distribution of RCMM	176
12.9.2.5	RCMM Processing.	176
12.9.2.6	Example: Renewal of a root certificate	177
12.9.3	Test certificates	177
12.10	Security on the return channel.	177
12.10.1	MHP functionality	177
12.10.2	TLS cipher suites.	178
12.10.3	Downloading of certificates for TLS.	178
12.10.3.1	Introduction	178
12.10.3.2	Usage of certificate in TLS	178
12.10.3.2.1	When certificates are delivered with the application	178
12.10.3.2.2	When no certificates are provided	179
12.11	The internet profile of X.509 (informative)	179
12.11.1	Main part of the certificate	179
12.11.1.1	Certificate.	179
12.11.1.2	signatureAlgorithm	179
12.11.1.3	signatureValue	180
12.11.1.4	tbsCertificate	180
12.11.1.5	version	180
12.11.1.6	serialNumber	181
12.11.1.7	signature.	181
12.11.1.8	issuer	181
12.11.1.9	validity	181
12.11.1.9.1	UTCTime	182
12.11.1.9.2	GeneralizedTime	182
12.11.1.10	subject	182
12.11.1.10.1	issuerUniqueID	182
12.11.1.10.2	subjectUniqueID	182
12.11.1.11	SubjectPublic Key Info	183
12.11.1.12	Unique Identifiers	183
12.11.1.13	Extensions	183
12.11.2	Standard certificate extensions	184
12.11.2.1	Authority key identifier	184
12.11.2.2	Subject key identifier.	184

12.11.2.3	Key usage	184
12.11.2.4	Private key usage period	184
12.11.2.5	Certificate policies	184
12.11.2.6	Policy mappings	185
12.11.2.7	Subject Alternative Name	185
12.11.2.8	Issuer Alternative Name	185
12.11.2.9	Subject Directory attributes	185
12.11.2.10	Basic Constraints	186
12.11.2.11	Name Constraints	186
12.11.2.12	Policy Constraints	186
12.11.2.13	Extended key usage field	186
12.11.2.14	CRL Distribution points	186
12.12	Platform minima	187
13	Graphics reference model	188
13.1	Introduction	188
13.1.1	Interapplication interaction	188
13.2	General Issues	189
13.2.1	Coordinate Spaces	189
13.2.1.1	Normalised screen space	189
13.2.1.2	User space	190
13.2.1.3	Pixel Aspect Ratio	192
13.2.1.4	Video space	193
13.3	Graphics	193
13.3.1	Modelling of the MHP display stack composition	193
13.3.2	AWT Reference Model in the MHP	196
13.3.3	HAVi devices and AWT components	196
13.3.3.1	Video and graphics pixel aligned	198
13.3.3.2	Zero graphics impact	198
13.3.4	Composition	198
13.3.4.1	AWT paint rule	198
13.3.5	Composition Rules	199
13.3.5.1	Components generally	199
13.3.6	Extensions to the AWT graphics capabilities	200
13.3.6.1	Graphics Objects in the MHP	200
13.3.6.2	Buffered Image	200
13.3.6.3	DVBColor	200
13.3.6.3.1	Modified packed colour representation	201
13.4	Video	201
13.4.1	Component-based players and background players	201
13.4.2	Modelling MPEG decoding and presentation pipeline	201
13.4.3	Coordinate Spaces	202
13.4.4	Video components	203
13.5	Subtitles	204
13.5.1	Language and presentation setting	204
13.5.2	Relation to graphics	204
13.5.3	Coordinate Spaces	204
13.6	Approximations	205
13.6.1	Approximations in composition	205
13.6.1.1	Implementation of modes	205
13.6.1.1.1	Graphics directly over video	205
13.6.1.1.2	Graphics over other graphics	205
13.6.1.2	Approximation of alpha	207
13.6.1.3	Approximation of colour	207
14	System integration aspects	208
14.1	Namespace mapping (DVB Locator)	208
14.1.1	dvb_entity = dvb_service	208
14.1.2	dvb_entity = dvb_service_component	209
14.1.3	dvb_hier_part = dvb_abs_path	209

14.1.4	dvb_abs_path	209
14.2	Reserved names	209
14.3	XML notation	210
14.4	Network signalling	212
14.5	Text encoding of application identifiers	212
14.6	Reserved names for persistent storage	212
14.7	Files and file names	213
14.8	Locators and content referencing	213
14.9	Service identification	214
14.9.1	Syntax of the textual service identifier	214
14.9.2	Handling of the textual service identifiers within the MHP terminal	215
14.10	CA system	215
14.10.1	Service selection	215
14.10.2	Media component selection	215
14.10.3	Non-media component selection	215
15	Detailed platform profile definitions	216
15.1	PNG - restrictions	218
15.1.1	PNG Aspect ratios	218
15.2	Minimum media formats supported by DVB-J APIs	218
15.3	JPEG - restrictions	218
15.4	Locale support	219
15.5	Video raster format dependencies	219
15.5.1	25 Hz standard definition	219
15.5.1.1	Logical pixel resolution	219
16	Registry of Constants	220
16.1	System constants	220
16.2	DVB-J constants	220
16.2.1	Public and Protected final static primitive fields from DVB packages	220
16.2.2	Public and Protected final static primitive fields from standard Java packages	223
Annex A (normative): External references; errata, clarifications and exemptions.		224
A.1	JAE 1.1.8 API [31]	224
A.1.1	java.lang.ThreadGroup.getParent()	224
A.1.2	java.net.URLConnection.setFileNameMap	224
A.1.3	java.util.Locale.setDefault	224
A.1.4	java.lang.Class	224
A.1.5	java.awt.Font	224
A.1.6	java.io.PrintStream	224
A.1.7	java.io.Serializable	224
A.1.8	java.io.ObjectStreamConstants	224
A.1.9	java.net.SocketOptions	225
A.1.10	java.util.zip.ZipConstants	225
A.1.11	Component	225
A.1.12	java.awt.event.KeyEvent	225
A.1.13	java.awt.Component	225
A.1.13.1	java.awt.Component.update(Graphics)	225
A.1.13.2	java.awt.Component.repaint()	225
A.1.13.3	java.awt.Component.repaint(long)	226
A.1.13.4	java.awt.Component.repaint(int, int, int, int)	226
A.2	Java Language Spec [32]	226
A.2.1	java.lang.ThreadGroup.getParent()	226
A.2.2	java.lang.Runtime.runFinalizersOnExit()	226
A.2.3	java.lang.System.runFinalizersOnExit()	226
A.3	Java Media Player Specification [33]	226
A.3.1	javax.media.protocol.URLDataSource.sources	226
A.3.2	javax.media.protocol.ContentDescriptor	226
A.3.2.1	getContentType	226
A.3.2.2	mimeTypeToPackageName	226

A.4	Java VM [34].	227
A.5	Java TV [51]	227
A.5.1	javax.tv.service.selection.	227
A.5.1.1	PresentationTerminatedEvent	227
A.5.1.2	ServiceContext.select(Locator [])	227
A.5.2	javax.tv.util.TVTimer	228
A.5.2.1	scheduleTimerSpec(TVTimerSpec)	228
A.5.3	javax.tv.util.TVTimerSpec	228
A.5.3.1	setAbsoluteTime(long)	228
A.5.3.2	setTime(long).	228
A.5.4	javax.tv.xlet.Xlet	228
A.5.4.1	Xlet state descriptions	228
A.5.4.2	Own service context.	228
A.5.5	javax.tv.graphics.AlphaColor	228
A.5.6	javax.tv.media.MediaSelectControl	229
A.5.6.1	addMediaSelectListener.	229
A.5.7	javax.tv.graphics.TVContainer	229
A.5.7.1	getRootContainer	229
A.6	DAVIC 1.4.1p9 [3]	229
A.6.1	org.davic.mpeg	229
A.6.1.1	General.	229
A.6.1.2	NotAuthorizedException	229
A.6.2	Chapter 9, Application Format	231
A.6.2.1	Section 9.4.7. "The MPEG-2 Section Filter API"	231
A.6.3	org.davic.mpeg.dvb	232
A.6.3.1	General.	232
A.6.4	org.davic.mpeg.sections	232
A.6.4.1	RingSectionFilter.	232
A.6.4.2	Section	232
A.6.4.2.1	clone()	232
A.6.4.2.2	getData().	232
A.6.4.2.3	getFullStatus()	232
A.6.4.3	SectionFilter.	232
A.6.4.3.1	Cross reference error	232
A.6.4.3.2	startFiltering(all signatures)	232
A.6.4.3.3	startFiltering(java.lang.Object, int, int, int, byte[], byte[]).	232
A.6.4.3.4	startFiltering (appData, pid, tableId) exceptions	233
A.6.4.3.5	Started Section Filters	233
A.6.4.4	SectionFilterGroup.	233
A.6.4.4.1	attach	233
A.6.4.4.2	Constructors	233
A.6.4.4.3	sectionSize	233
A.6.4.4.4	newRingSectionFilter.	233
A.6.4.4.5	Constructor(int, boolean).	233
A.6.4.5	TimeOutEvent	233
	TimeOutEvent	234
A.6.5	org.davic.media	235
A.6.5.1	FreezeControl.resume()	235
A.6.5.2	MediaTimePositionChangedEvent	235
A.6.5.3	NotAuthorizedMediaException	235
A.6.5.4	LanguageControl	236
A.6.6	org.davic.net	236
A.6.6.1	InvalidLocatorException	236
A.6.6.2	Locator	237
A.6.6.2.1	Locator().	237
A.6.6.2.2	toExternalForm()	237
A.6.6.3	tuning	237
A.6.6.3.1	NetworkInterfaceController	237
A.6.6.3.2	NetworkInterface	238

A.6.6.4	ca	238
A.6.6.4.1	CAMessage	238
A.6.6.4.2	CAModule	239
A.6.6.4.3	CAModuleManager	242
A.6.6.4.4	NoFreeCapacityException	242
A.6.6.4.5	MMIOObject	243
A.6.6.4.6	DescramblerProxy	244
A.6.6.4.7	StartMMIEvent(MMIOObject, int, java.lang.Object)	245
A.6.6.4.8	ModuleResponseEvent	245
A.6.6.4.9	NewModuleEvent	245
A.6.6.4.10	PIDChangeEvent	245
A.6.6.4.11	TuneRequestEvent	246
A.6.6.5	dvb.DvbLocator(int onid, int tsid, int serviceid, int eventid, int componenttags[], String filePath)	247
A.6.6.6	dvb.DvbLocator	247
A.6.6.6.1	DvbLocator(int, int, int, int, int[])	247
A.6.6.6.2	Additional method	247
A.6.6.6.3	getOriginalNetworkId	247
A.6.7	org.davic.net.tuning	247
A.6.7.1	Figure H-1	247
A.6.7.2	Figure H-2	247
A.6.8	Extensibility and Over-Riding	248
A.7	HAVi [50]	248
A.7.1	Drafting conventions	248
A.7.2	General	248
A.7.2.1	Thread-safety	248
A.7.3	Event mechanism	248
A.7.3.1	Introduction	248
A.7.3.2	Overview of HAVi events	249
A.7.3.3	Relation between HAVi events and AWT events	250
A.7.3.4	Application guidelines	250
A.7.4	org.havi.ui	250
A.7.4.1	HActionable	250
A.7.4.1.1	getActionSound	251
A.7.4.2	HActionInputPreferred	251
A.7.4.3	HAdjustmentInputPreferred	251
A.7.4.3.1	Interface description	251
A.7.4.3.2	getAdjustMode	251
A.7.4.3.3	processHAdjustmentEvent	251
A.7.4.3.4	setAdjustMode	251
A.7.4.4	HAdjustmentValue	251
A.7.4.4.1	Class description	251
A.7.4.4.2	getAdjustmentSound	252
A.7.4.4.3	getBlockIncrement	252
A.7.4.4.4	getUnitIncrement	252
A.7.4.5	HAnimateEffect	252
A.7.4.5.1	getRepeatCount	252
A.7.4.6	HBackgroundDevice	252
A.7.4.7	HBackgroundImage	252
A.7.4.7.1	Constructors - HBackgroundImage(String), HBackgroundImage(URL)	252
A.7.4.7.2	load	252
A.7.4.8	HComponent	252
A.7.4.8.1	processEvent	252
A.7.4.9	HComponentOrdering	253
A.7.4.9.1	addAfter, addBefore	253
A.7.4.10	HEventMulticaster	253
A.7.4.10.1	Class description	253
A.7.4.10.2	Constructor	253
A.7.4.10.3	remove	253

A.7.4.11	HFontCapabilities	253
A.7.4.11.1	getSupportedCharacterRanges	253
A.7.4.12	HGraphicsDevice	253
A.7.4.12.1	getBestConfiguration	253
A.7.4.12.2	setGraphicsConfiguration	254
A.7.4.13	HGraphicsConfigTemplate	254
A.7.4.13.1	setPreference	254
A.7.4.14	HListElement	254
A.7.4.14.1	Class description	254
A.7.4.15	HListGroup	254
A.7.4.15.1	Class description	254
A.7.4.15.2	Fields	255
A.7.4.15.3	Use of "action" and "actioning"	255
A.7.4.15.4	addItem(HListItem item, int index)	255
A.7.4.15.5	addItem(HListItem items[], int index)	255
A.7.4.15.6	getIconSize	255
A.7.4.15.7	getLabelSize	255
A.7.4.15.8	getOrientation	256
A.7.4.15.9	setFocusTraversal	256
A.7.4.15.10	setItemSelected	256
A.7.4.15.11	setIconSize	256
A.7.4.15.12	setLabelSize	256
A.7.4.15.13	setListContent	256
A.7.4.15.14	setMove	256
A.7.4.15.15	setScrollPosition	256
A.7.4.15.16	setSelectionMode	257
A.7.4.16	HListGroupLook	257
A.7.4.16.1	getMaximumSize	257
A.7.4.16.2	getMinimumSize	257
A.7.4.16.3	getPreferredSize	257
A.7.4.16.4	getValue	258
A.7.4.16.5	hitTest	258
A.7.4.16.6	showLook	258
A.7.4.17	HLook	260
A.7.4.17.1	General	260
A.7.4.17.2	Class description	260
A.7.4.17.3	getPreferredSize	260
A.7.4.17.4	showLook	260
A.7.4.18	HMultilineEntry	260
A.7.4.19	HMultilineEntryLook	260
A.7.4.19.1	getCaretCharPositionForLine	260
A.7.4.19.2	getSoftLineBreakPositions	261
A.7.4.19.3	getVisibleSoftLineBreakPositions	261
A.7.4.20	HNavigable	261
A.7.4.20.1	Class description	261
A.7.4.21	HOrientable	261
A.7.4.21.1	Interface description	261
A.7.4.21.2	getOrientation	261
A.7.4.21.3	setOrientation	261
A.7.4.22	HScene	262
A.7.4.22.1	addAfter, addBefore	262
A.7.4.22.2	getFocusOwner	262
A.7.4.23	HScreenConfigurationListener	262
A.7.4.24	HSceneFactory	262
A.7.4.24.1	getDefaultHScene	262
A.7.4.25	HSelectionInputPreferred	262
A.7.4.25.1	Interface description	262
A.7.4.25.2	getSelectionMode	263
A.7.4.25.3	processHItemEvent	263

A.7.4.25.4	setSelectionMode	263
A.7.4.26	HSingleLineEntry	263
A.7.4.26.1	Class description	263
A.7.4.27	HStaticAnimation	263
A.7.4.28	HStaticRange	263
A.7.4.28.1	Fields	263
A.7.4.28.2	getOrientation	263
A.7.4.28.3	setBehavior	264
A.7.4.28.4	setRange	264
A.7.4.28.5	setThumbOffsets	264
A.7.4.28.6	setValue	264
A.7.4.29	HVideoDevice	264
A.7.4.29.1	getBestConfiguration	264
A.7.4.29.2	setVideoConfiguration	264
A.7.4.30	HVisible	264
A.7.4.30.1	Class description	264
A.7.4.30.2	Constructors	265
A.7.4.30.3	setDefaultSize	265
A.7.4.30.4	setLookData	265
A.7.4.30.5	setResizeMode	265
A.7.4.30.6	setTextContent	265
A.7.5	org.havi.ui.event	265
A.7.5.1	HActionEvent	265
A.7.5.2	HItemEvent	266
A.7.5.2.1	Constructor	266
A.7.5.3	HKeyEvent	266
A.7.5.3.1	Constructors	266
A.7.5.4	HRcEvent	266
A.7.5.5	HRcCapabilities	266
A.7.6	References to ISO/IEC10646-1:1993	266
A.8	ISO/IEC 13818-6 [26]	266
A.8.1	Reconstruction of NPT	266
Annex B (normative): Object carousel		267
B.1	Introduction	267
B.1.1	Key to notation	267
B.2	Object Carousel Profile	267
B.2.1	DSM-CC Sections	267
B.2.1.1	Sections per TS packet	268
B.2.2	Data Carousel	268
B.2.2.1	General	268
B.2.2.2	DownloadInfoIndication	268
B.2.2.3	DownloadServerInitiate	268
B.2.2.4	ModuleInfo	269
B.2.2.4.1	Label descriptor	269
B.2.2.4.2	Caching priority descriptor	270
B.2.2.5	ServiceGatewayInfo	271
B.2.2.6	Download Cancel	271
B.2.3	The Object Carousel	272
B.2.3.1	BIOP Generic Object Message	272
B.2.3.2	CORBA strings	272
B.2.3.3	BIOP FileMessage	272
B.2.3.4	Content type descriptor	273
B.2.3.5	BIOP DirectoryMessage	274
B.2.3.6	BIOP ServiceGateway message	276
B.2.3.7	BIOP Interoperable Object References	276
B.2.3.7.1	BIOPProfileBody	277
B.2.3.7.2	LiteOptionsProfileBody	278
B.2.3.8	BIOP StreamMessage	281

B.2.3.9	BIOP StreamEventMessage	283
B.2.4	Stream Events	284
B.2.4.1	Stream & StreamEvent messages	284
B.2.4.1.1	Association with time bases	284
B.2.4.1.2	Event names and event ids	285
B.2.4.1.3	Stream event life time	285
B.2.4.2	Stream Descriptors	285
B.2.4.2.1	NPT Reference descriptor	285
B.2.4.2.2	Stream event descriptor	286
B.2.4.2.3	Unused descriptors	287
B.2.4.2.4	Clarification of number encoding	287
B.2.4.3	DSM-CC Sections carrying Stream Descriptors	287
B.2.4.3.1	Section version number	287
B.2.4.3.2	Single firing of "do it now" events	287
B.2.4.3.3	Section number	287
B.2.4.3.4	DSM-CC sections for DSMCC_descriptor_list()	287
B.2.4.3.5	Encoding of table id extension	287
B.2.4.4	Timebases	288
B.2.4.5	Monitoring stream events	289
B.2.4.5.1	NPT reference monitoring	289
B.2.4.5.2	Timebase stimulated event monitoring	289
B.2.4.5.3	"do it now" events	289
B.2.4.5.4	scheduled events	289
B.2.4.5.5	number of NPT components	289
B.2.5	Assignment and use of transactionId values	290
B.2.6	Informative Background	290
B.2.7	DVB semantics of the transactionId field	290
B.2.8	Mapping of objects to data carousel modules	291
B.2.9	Compression of modules	291
B.2.10	Mounting an Object Carousel	292
B.2.10.1	carousel_id_descriptor	292
B.2.10.2	DVB-J mounting of an object carousel	293
B.2.11	Unavailability of a carousel	294
B.2.12	Delivery of Carousel within multiple services	294
B.3	AssociationTag Mapping	295
B.3.1	Decision algorithm for association tag mapping	295
B.3.1.1	TapUse is not BIOP_PROGRAM_USE	295
B.3.1.2	TapUse is BIOP_PROGRAM_USE	296
B.3.2	DSM-CC association_tags to DVB component_tags	296
B.3.3	deferred_association_tags_descriptor	296
B.4	Example of an Object Carousel (informative)	297
B.5	Caching	298
B.5.1	Determining file version	298
B.5.2	Transparency levels of caching	298
B.5.2.1	Transparent caching	298
B.5.2.1.1	Active caching	299
B.5.2.1.2	Passive caching	299
B.5.2.1.3	DII repetition rate	299
B.5.2.2	Semi-transparent caching	299
B.5.2.2.1	Implications for the terminal (informative)	299
B.5.2.3	Static caching	300
B.5.2.3.1	Implications for the broadcaster (informative)	300
B.5.2.3.2	Implications for the terminal (informative)	300
Annex C (informative):	References	301
Annex D (normative):	Text presentation	302
D.1	Scope	302
D.2	Fonts	302
D.2.1	Embedded fonts	302

D.2.2	Downloaded fonts	302
D.2.2.1	Font technology	302
D.2.2.2	Font index files	303
D.2.2.2.1	Format of file	303
D.2.2.2.2	Element semantics	303
D.2.2.2.3	Example	304
D.2.2.3	Name and location of font index files	304
D.2.2.3.1	General	304
D.2.2.3.2	Name of file	304
D.2.2.3.3	Location	304
D.2.2.4	Specification of fonts at run time	304
D.2.2.4.1	DVB-J	304
D.3	Text rendering	305
D.3.1	Low and high level rendering	305
D.3.1.1	Low level rendering	305
D.3.1.2	High level rendering	305
D.3.2	Philosophy	305
D.3.2.1	High level rendering conceptual process	305
D.3.3	Font Definition	306
D.3.3.1	Font bounds	306
D.3.3.2	"Physical" font data	307
D.3.4	Converting font metrics to display pixels	307
D.3.4.1	Vertical resolution	307
D.3.4.2	Horizontal resolution	307
D.3.5	Rendering within limits and insets	308
D.3.5.1	Low level rendering	308
D.3.5.2	High level rendering	308
D.3.5.3	Conversion of units	308
D.3.6	"logical" text width rules	309
D.3.6.1	Computing "logical" text width	309
D.3.6.1.1	Font sizes	309
D.3.6.1.2	Character widths	309
D.3.6.1.3	Kerning	310
D.3.6.1.4	Letter spacing	310
D.3.6.2	Logical text width	310
D.3.7	Line breaking	310
D.3.7.1	Text wrapping setting is false	310
D.3.7.2	Text wrapping setting is true	311
D.3.8	Positioning lines of text vertically within an object	311
D.3.8.1	Number of lines	311
D.3.8.2	Truncation	312
D.3.8.3	Positioning	312
D.3.8.3.1	Vertical alignment setting is VERTICAL_START_ALIGN	312
D.3.8.3.2	Vertical alignment setting is VERTICAL_END_ALIGN	312
D.3.8.3.3	Vertical alignment setting is VERTICAL_CENTER	313
D.3.8.3.4	Examples	314
D.3.9	Rendering lines of text horizontally	314
D.3.9.1	Available width	314
D.3.9.2	Truncation	314
D.3.9.3	Placement	315
D.3.10	Text overflow	315
D.3.11	Tabulation	316
D.3.12	Placing runs of characters & words	316
D.3.13	Control of text flow	317
D.4	Text mark-up	317
D.4.1	White Space Characters	317
D.4.2	Marker characters	318
D.4.3	Non-printing characters	318
D.4.4	Format Control Mark-up	318

D.4.5	Future compatibility	319
Annex E (normative): Character set		
E.1	Basic Euro Latin character set	320
Annex F (informative): Authoring & Implementation Guidelines		
F.1	Authoring Guidelines	325
F.2	Implementation Guidelines	325
F.3	Authoring guidelines for DVB-J	325
Annex G (normative): Minimum Platform Capabilities		
G.1	Graphics	326
G.1.1	Device capabilities	326
G.1.2	Video presentation capabilities	326
G.1.3	Image processing capabilities	327
G.1.3.1	Composition rules	327
G.1.4	Alpha capabilities	327
G.1.5	Colour capabilities	327
G.1.6	MPEG I frame and Video drips	328
G.2	Audio	328
G.3	Video	329
G.4	Resident fonts and text rendering	329
G.4.1	The built-in font	329
G.4.2	Presentation to DVB-J	329
G.4.3	Text directions	329
G.5	Input events	330
G.6	Memory	330
G.7	Other resources	331
Annex H (normative): Extensions		
Annex I (normative): DVB-J fundamental classes		
	org.dvb.lang	335
	DVBClassLoader	336
Annex J (normative): DVB-J event API		
J.1	Overview	338
J.2	The resource management	340
J.3	The Event Repository	340
J.3.1	Example	340
J.4	Unicode	341
J.5	Virtual keyboards	341
	org.dvb.event	342
	EventManager	343
	OverallRepository	346
	RepositoryDescriptor	347
	UserEvent	348
	UserEventAvailableEvent	352
	UserEventListener	354
	UserEventRepository	355
	UserEventUnavailableEvent	358
Annex K (normative): DVB-J persistent storage API		
	org.dvb.io.persistent	361
	FileAccessPermissions	362
	FileAttributes	364
Annex L (normative): User Settings and Preferences API		
	org.dvb.user	368
	Facility	369
	GeneralPreference	370

	Preference	371
	UnsupportedPreferenceException	374
	UserPreferenceChangeEvent	375
	UserPreferenceChangeListener	376
	UserPreferenceManager	377
	UserPreferencePermission	379
Annex M (normative):	SI Access API	380
M.1	Unicode	380
	org.dvb.si	381
	Descriptor	384
	DescriptorTag	386
	PMTElementaryStream	390
	PMTService	392
	PMTStreamType	394
	SIBouquet	395
	SIDatabase	399
	SIEvent	416
	SIException	420
	SIIllegalArgumentException	421
	SIInformation	422
	SIInvalidPeriodException	426
	SIIterator	427
	SILackOfResourcesEvent	428
	SIMonitoringEvent	429
	SIMonitoringListener	432
	SIMonitoringType	433
	SINetwork	434
	SINotInCacheEvent	438
	SIOBJECTNotInTableEvent	439
	SIRequest	440
	SIRequestCancelledEvent	441
	SIRetrievalEvent	442
	SIRetrievalListener	444
	SIRunningStatus	445
	SIService	446
	SIServiceType	452
	SISuccessfulRetrieveEvent	454
	SITableNotFoundEvent	455
	SITableUpdatedEvent	456
	SITime	457
	SITransportStream	458
	SITransportStreamBAT	460
	SITransportStreamDescription	461
	SITransportStreamNIT	462
	SIUtil	463
	TextualServiceIdentifierQuery	464
Annex N (normative):	Streamed Media API Extensions	465
	org.dvb.media	466
	ActiveFormatDescriptionChangedEvent	468
	AspectRatioChangedEvent	469
	BackgroundVideoPresentationControl	470
	CAException	471
	CAStopEvent	472
	DFCChangedEvent	474
	DripFeedDataSource	475
	DripFeedPermission	478
	DVBMediaSelectControl	480
	NoComponentSelectedEvent	481

PresentationChangedEvent	483
ServiceRemovedEvent	485
StopByResourceLossEvent	487
SubtitleAvailableEvent	489
SubtitleListener	490
SubtitleNotAvailableEvent	491
SubtitleNotSelectedEvent	492
SubtitleSelectedEvent	493
SubtitlingEventControl	494
VideoFormatControl	495
VideoFormatEvent	500
VideoFormatListener	501
VideoPresentationControl	502
VideoTransformation	507

Annex O (normative): Integration of the JavaTV SI API and DVB SI 510

O.1	Introduction	510
O.2	Mapping of the JavaTV SI API to DVB SI	510
O.2.1	javax.tv.service.Service	510
O.2.1.1	getName	510
O.2.1.2	getServiceType	510
O.2.2	javax.tv.service.ServiceComponent	510
O.2.2.1	getComponentName	510
O.2.2.2	getAssociatedLanguage	510
O.2.2.3	getStreamType	511
O.2.3	javax.tv.service.ServiceType	511
O.2.4	javax.tv.service.StreamType	511
O.2.5	javax.tv.service.SIElement	511
O.2.5.1	getServiceInformationType	512
O.2.6	javax.tv.service.SIManager	512
O.2.6.1	getSupportedDimensions	512
O.2.6.2	getRatingDimension	512
O.2.6.3	retrieveSIElement	512
O.2.6.4	getTransports	512
O.2.6.5	filterServices	512
O.2.7	javax.tv.service.navigation.SIElementFilter	512
O.2.8	javax.tv.service.navigation.ServiceDetails	512
O.2.8.1	getLongName	512
O.2.8.2	getServiceType	512
O.2.8.3	retrieveServiceDescription	512
O.2.8.4	retrieveComponents	513
O.2.9	javax.tv.service.navigation.CAIdentification	513
O.2.9.1	getCASystemIds	513
O.2.9.2	isFree	513
O.2.10	javax.tv.service.RatingDimension	513
O.2.10.1	getDimensionName	513
O.2.10.2	getNumberOfLevels	513
O.2.10.3	getRatingLevelDescription	513
O.2.11	javax.tv.service.navigation.ServiceProviderInformation	513
O.2.11.1	getProviderName	513
O.2.12	javax.tv.service.transport.Transport	513
O.2.13	javax.tv.service.transport.Bouquet	514
O.2.13.1	getBouquetID	514
O.2.13.2	getName	514
O.2.13.3	getLocator	514
O.2.14	javax.tv.service.transport.Network	514
O.2.14.1	getNetworkID	514
O.2.14.2	getName	514
O.2.14.3	getLocator	514

O.2.15	javax.tv.service.transport.TransportStream	514
O.2.15.1	getTransportStreamID	514
O.2.15.2	getDescription	514
O.2.16	javax.tv.service.guide.ProgramEvent	514
O.2.16.1	getDuration	514
O.2.16.2	getStartTime	515
O.2.16.3	getEndTime	515
O.2.16.4	getName	515
O.2.16.5	retrieveDescription	515
O.2.16.6	getRating	515
O.2.17	javax.tv.service.guide.ContentRatingAdvisory	515
O.2.17.1	getDimensionNames	515
O.2.17.2	getRatingLevel	515
O.2.17.3	getRatingText	515
O.2.17.4	getDisplayText	515
O.3	Integration of the JavaTV SI API and the DVB SI API	516

Annex P (normative): Broadcast Transport Protocol Access 517

org.dvb.dsmcc	519
AsynchronousLoadingEvent	521
AsynchronousLoadingEventListener	522
DSMCCException	523
DSMCCObject	524
DSMCCStream	531
DSMCCStreamEvent	535
IllegalObjectTypeException	538
InsufficientResourcesException	539
InvalidAddressException	540
InvalidFormatEvent	541
InvalidFormatException	542
InvalidPathnameEvent	543
InvalidPathNameException	544
LoadingAbortedEvent	545
MPEGDeliveryErrorEvent	546
MPEGDeliveryException	547
NotEntitledEvent	548
NotEntitledException	549
NothingToAbortException	550
NotLoadedException	551
NPDiscontinuityEvent	552
NPListener	554
NPPresentEvent	555
NPTRate	556
NPTRateChangeEvent	557
NPTRemovedEvent	559
NPStatusEvent	560
ObjectChangeEvent	561
ObjectChangeListener	563
ServerDeliveryErrorEvent	564
ServerDeliveryException	565
ServiceDomain	566
ServiceXFRErrorEvent	571
ServiceXFRException	573
ServiceXFRReference	575
StreamEvent	577
StreamEventListener	579
SuccessEvent	580
UnknownEventException	581

Annex Q (normative):	Datagram Socket Buffer Control	582
	org.dvb.net.	583
	DatagramSocketBufferControl	584
Annex R (normative):	DVB-J Return Channel Connection Management API	585
	org.dvb.net.rc	586
	ConnectionEstablishedEvent	587
	ConnectionFailedEvent	588
	ConnectionListener	589
	ConnectionParameters	590
	ConnectionRCEvent	592
	ConnectionRCInterface	593
	ConnectionTerminatedEvent	597
	IncompleteTargetException	598
	PermissionDeniedException	599
	RCInterface	600
	RCInterfaceManager	602
	RCInterfaceReleasedEvent	604
	RCInterfaceReservedEvent	605
	RCPermission	606
Annex S (normative):	Application Listing and Launching	608
	org.dvb.application	609
	AppAttributes	611
	AppIcon	616
	AppID	617
	AppProxy	619
	AppsControlPermission	623
	AppsDatabase	625
	AppsDatabaseEvent	629
	AppsDatabaseEventListener	631
	AppsDatabaseFilter	633
	AppStateChangeEvent	634
	AppStateChangeEventListener	636
	CurrentServiceFilter	637
	DVBHTMLProxy	638
	DVBJProxy	640
	IllegalProfileParameterException	642
	LanguageNotAvailableException	643
	RunningApplicationsFilter	644
Annex T (normative):	Permissions	645
	org.dvb.net.ca	646
	CAPermission	647
	org.dvb.net.tuning	649
	DvbNetworkInterfaceSIUtil	650
	TunerPermission	651
Annex U (normative):	Extended graphics APIs	653
	org.dvb.ui	654
	DVBAlphaComposite	655
	DVBBufferedImage	660
	DVBColor	668
	DVBGraphics	672
	DVBRasterFormatException	676
	DVBTextLayoutManager	677
	FontFactory	684
	FontFormatException	686
	FontNotAvailableException	687
	TestOpacity	688

TextOverflowListener	689
UnsupportedDrawingOperationException	690
Annex V :	Void..... 691
Annex W (informative):	DVB-J examples 692
W.1 DVB-J Application lifecycle implementation example	692
W.2 Example of exporting an object for inter-application communication	693
W.3 Example of use of video drip feed	693
W.4 Example of CPU bound animation	695
Annex X (normative):	Test support 698
Broadcast model	699
DVBTest	701
Annex Y (normative):	Inter-application communication API..... 707
org.dvb.io.ixc	708
IxcRegistry	709
Annex Z (informative):	Services, Service Contexts and Applications in an MHP Environment 712
Z.1 Introduction.....	712
Z.2 Basic concepts.....	712
Z.3 Presenting a service in MHP.....	712
Z.3.1 Presenting the media components of a service	712
Z.3.2 Presenting the application components of a service	712
Z.4 Multiple service contexts in an MHP platform.....	713
Z.5 How does the platform know which services are available?	714
Index.....	715
History	752

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://www.etsi.org/legal/home.htm>).

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.

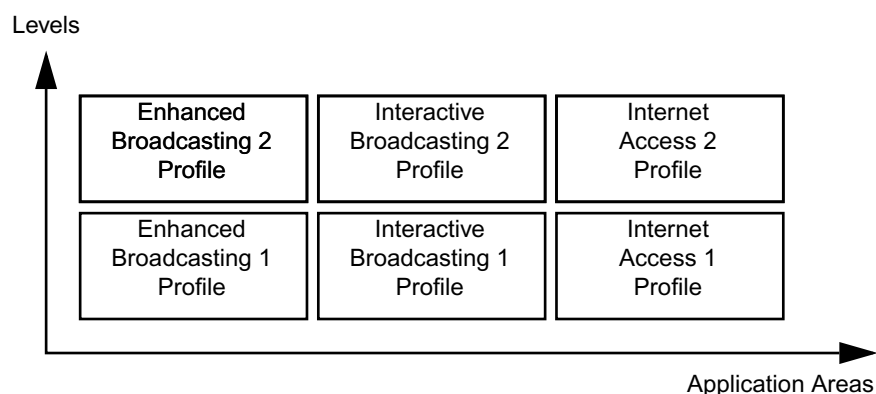


Figure 1 : Application areas and levels of profiles

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 224](#). 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 [4] are provided for convenience to access the document in electronic form.
[5]	URL references with note [5] are the normative method to access the reference

[6]	ETSI specifications are available from the ETSI server at: http://www.etsi.org . 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: http://java.sun.com/products/specformhp/ 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. ftp://ftp.omg.org/pub/docs/formal/97-09-01.pdf	[5]
[3]	DAVIC 1.4.1p9	June 1999	DAVIC 1.4.1 Specification Part 9, Complete DAVIC Specifications, DAVIC. http://www.davic.org	[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]	TR 101 162		Digital broadcasting systems for television, sound and data services; Allocation of Service Information (SI) codes for Digital Video Broadcasting (DVB) systems. The normative version of this document is available on-line from: http://www.dvb.org This document is also available in physical form from the DVB project office at: DVB Project Office, 17a Ancienne Route, CH-1218 Grand Saconnex, Geneva, Switzerland.	[5]
[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	1.2.1	Digital Video Broadcasting (DVB), DVB subtitling	[6]
[14]	ETS 300 800	1	DVB Interaction Channel for Cable TV distribution systems	[6]

Reference	Edition	Description	Note
[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 http://www.w3.org/Graphics/GIF/spec-gif89a.txt	
[18]	ISO 10646-1	Information technology - Universal multiple-octet coded character set (UCS), part 1: Architecture and Basic Multilingual Plane	[1]
[19]	ISO 639.2	Code for the representation of names of languages	[1]
[20]	ISO 8859-1	Information technology -- 8-bit single-byte coded graphic character sets -- Part 1: Latin alphabet No. 1	[1]
[21]	ISO/IEC 10918-1	Digital compression encoding of continuous-tone still images (JPEG). http://www.w3.org/Graphics/JPEG/itu-t81.pdf	[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 nd 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. ftp://ftp.javasoft.com/docs/specs/langspec-1.0.pdf including the clarifications at: http://java.sun.com/docs/books/jls/clarify.html	[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]

Reference	Edition	Description	Note
[35]	JFIF	JPEG File Interchange Format, Eric Hamilton, C-Cube Microsystems. http://www.w3.org/Graphics/JPEG/jfif3.pdf	[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 http://www.w3.org/TR/REC-png.html	[5]
[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 1.1	This comprises the following documents: <ul style="list-style-type: none"> • HAVi v1.1 Chapter 8, 15-May-2001 • HAVi v1.1 Java L2 APIs, 15-May-2001 available at: http://www.havi.org	
[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)"	

Reference	Edition	Description	Note
[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]
[62] FIPS-180-1	April 1995	NIST, FIPS PUB 180-1: Secure Hash Standard http://csrc.nist.gov/fips/fip180-1.txt	[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 http://www.w3.org/TR/2000/REC-xml-20001006	
[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 1035	November 1987	Domain names implementation and specification	
[76] RFC 1950	May 1996	ZLIB Compressed Data Format Specification version 3.3	
[77] RFC 1951	May 1996	DEFLATE Compressed Data Format Specification version 1.3	
[78] XHTML™ 1.0: The Extensible HyperText Markup Language		http://www.w3.org/TR/xhtml1	
[79] JDK 1.2.2	Part of ISBN:1-892488-25-6		
[80] R206		Guidelines for Implementation and Use of the Common Interface for DVB Decoder Applications	
[81] Tiresias		The RNIB/DTG "Tiresias"	

	Reference	Edition	Description	Note
[82]	TR 101 262	V1.1.1	ETSI drafting rules	
[83]	EN 301 790	V1.2.2	Digital Video Broadcasting (DVB); Interaction channel for satellite distribution systems;	

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 instance: A unique invocation of an application. i.e. running the same application twice results in two distinct application instances.

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.

best effort: an implementation dependent approximation which is as close as reasonable to what has requested in the circumstances concerned.

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.

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.

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 and extensions defined in future versions of this specification.

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.

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.

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.

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

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
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.

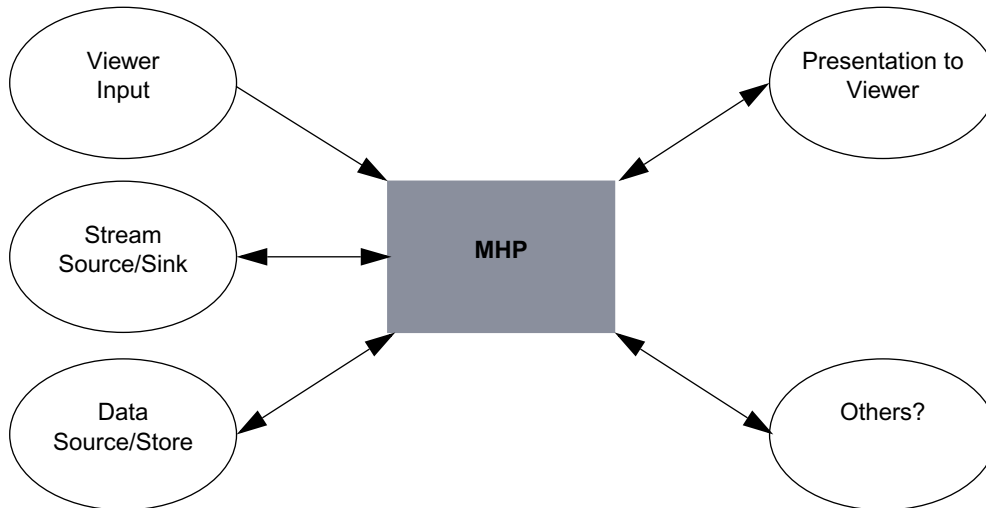


Figure 2 : MHP context

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.

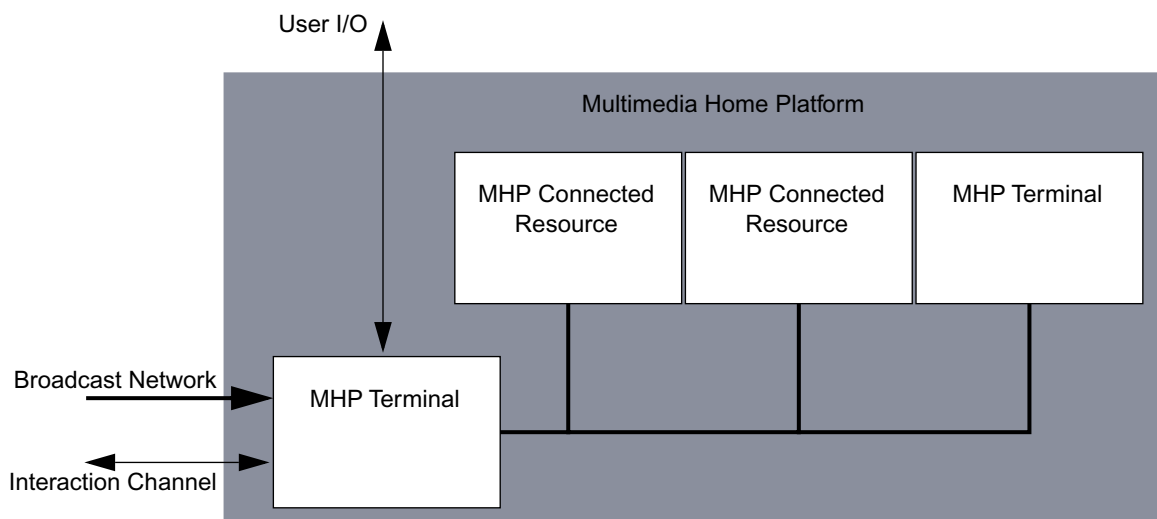


Figure 3 : External interfaces between an MHP and the outside world

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.

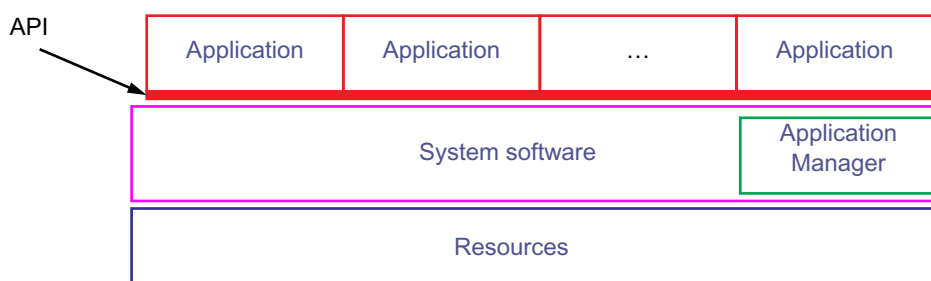


Figure 4 : Basic architecture

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 40 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.

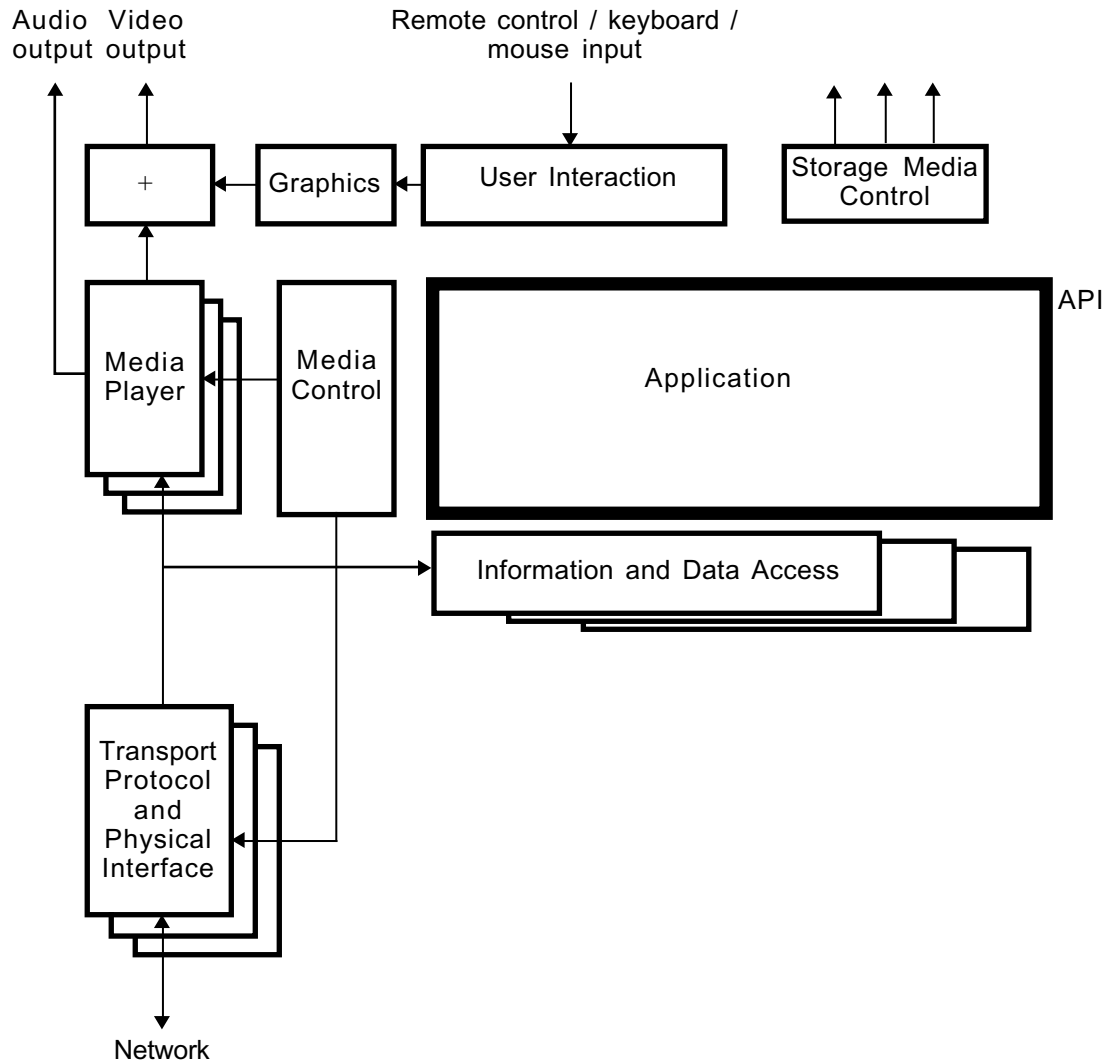


Figure 5 : Interfaces between an MHP application and the MHP system

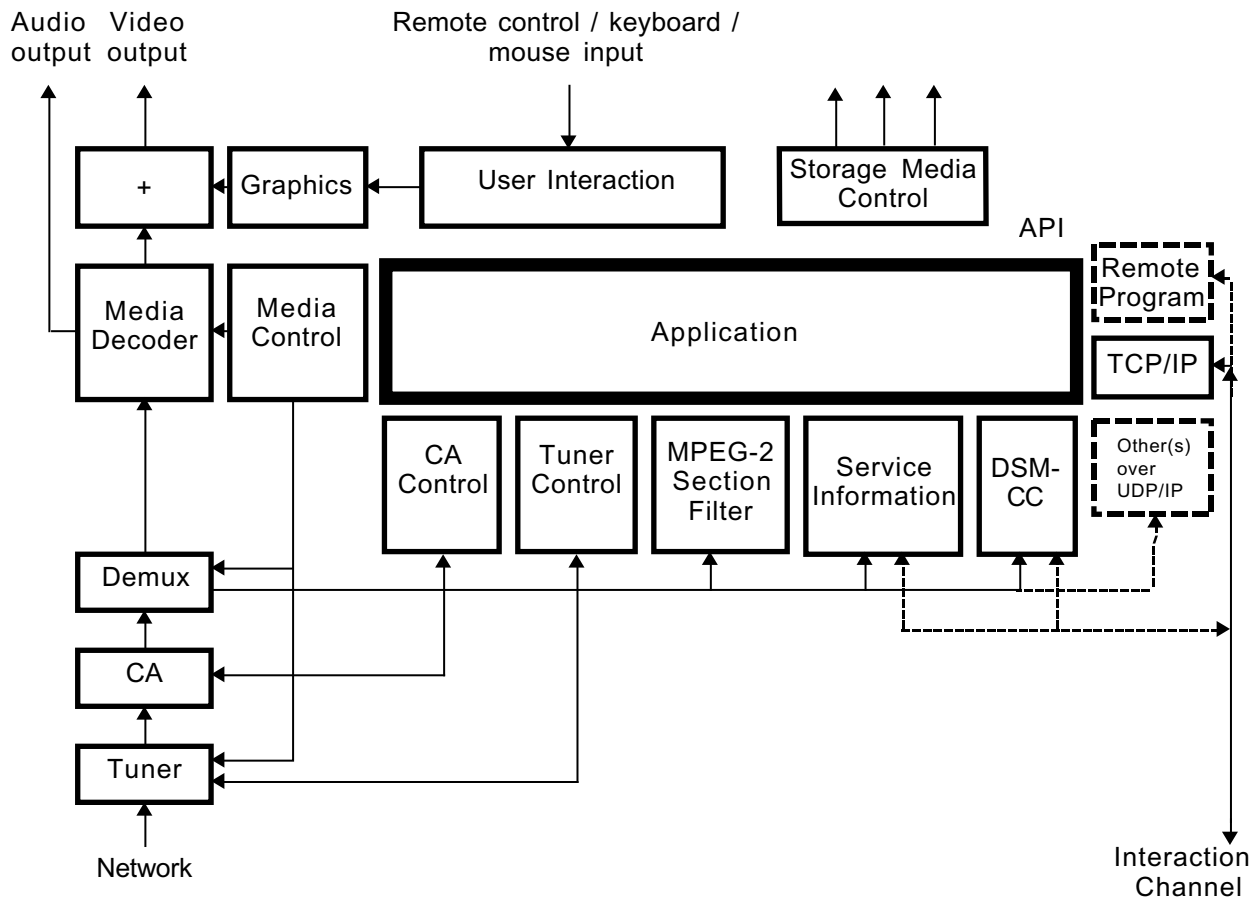


Figure 6 : Interfaces between an MHP application and the MHP system with more details

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.

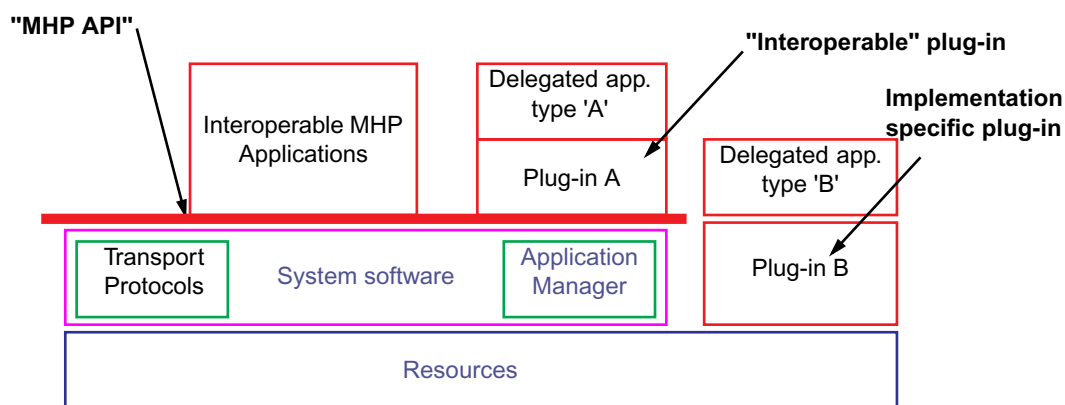


Figure 7 : Illustrative plug-in implementation options

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 \[48\]](#), [EN 301 790 \[83\]](#) respectively for CATV, PSTN/ISDN, DECT, GSM, LMDS, SMATV and satellite 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 [H, "\(normative\): Extensions" on page 333](#).

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 216). The full details of the APIs that provide access to these broadcast protocols are in chapter 11, "DVB-J Platform" on page 102.

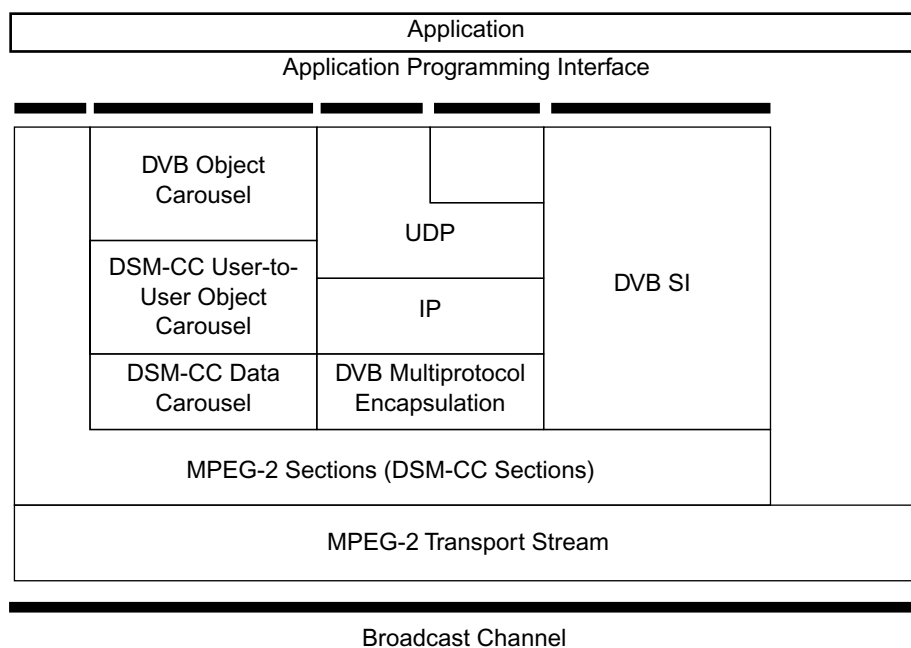


Figure 8 : Broadcast Channel Protocol Stack

Except in the case of MPEG-2 sections (see 6.2.2 "MPEG-2 Sections"), when an MHP application attempts to access conditional access scrambled data through one of these broadcast channel protocols, the MHP terminal shall attempt to initiate descrambling of this data without the application needing to explicitly ask for it. Attempts to access conditional access scrambled data at the level of MPEG-2 sections shall not happen without the application explicitly asking for this.

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 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 267.

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, carousel data streams servicing broadcast file systems may become unavailable. The conditions for permanent loss of carousel are defined in [B.2.11, "Unavailability of a carousel" on page 294](#). The conditions for temporary disconnection and reconnection of carousel are defined in [9.1.5, "Persistence of Applications Across Service Boundaries" on page 59](#).

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. Permanently lost carousels shall never be restored automatically. Temporary disconnections and reconnections are automatic and are largely invisible to the application. However, implementations must preserve Java IO semantics. For example, the `InputStream.available()` method should accurately report the number of bytes available without blocking.

Data not in such a cache shall be unavailable to applications. When applications attempt to access unavailable data from permanently lost carousels, the operation 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 carousel" on page 120](#).

When an application attempts to access unavailable data from a temporarily disconnected carousel, the operation shall block until the data becomes available, or it is interrupted. This includes any operations that indirectly cause synchronous loading operations, as stated in [11.5.1.1, "Constraints on the java.io.File methods for broadcast carousels" on page 119](#).

Upon reconnection to a carousel, the system shall start fetching any outstanding data. Any blocked I/O operations shall return once the data is received.

A temporarily disconnected service may become permanently lost if the system determines that the loss of connection is irrecoverable - as stated in [P, "\(normative\): Broadcast Transport Protocol Access" on page 517](#), org.dvb.dsmcc. `ServiceDomain`. The cases in which this may occur are:

- The carousel becomes permanently unavailable, as stated in [B.2.11, "Unavailability of a carousel" on page 294](#).
- The period since the carousel became disconnected is at least 60 seconds.

If this occurs, any blocked I/O operations shall terminate with `InterruptedException` and the `ServiceDomain` shall enter the detached state. An implementation may choose to use longer timeouts other strategies to determine when to permanently lose carousels, within the constraints listed above. For example, boot carousels for running applications may be kept in preference to carousels mounted by applications.

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 216). The full details of the APIs that provide access to these interaction protocols are in chapter 11, "DVB-J Platform" on page 102.

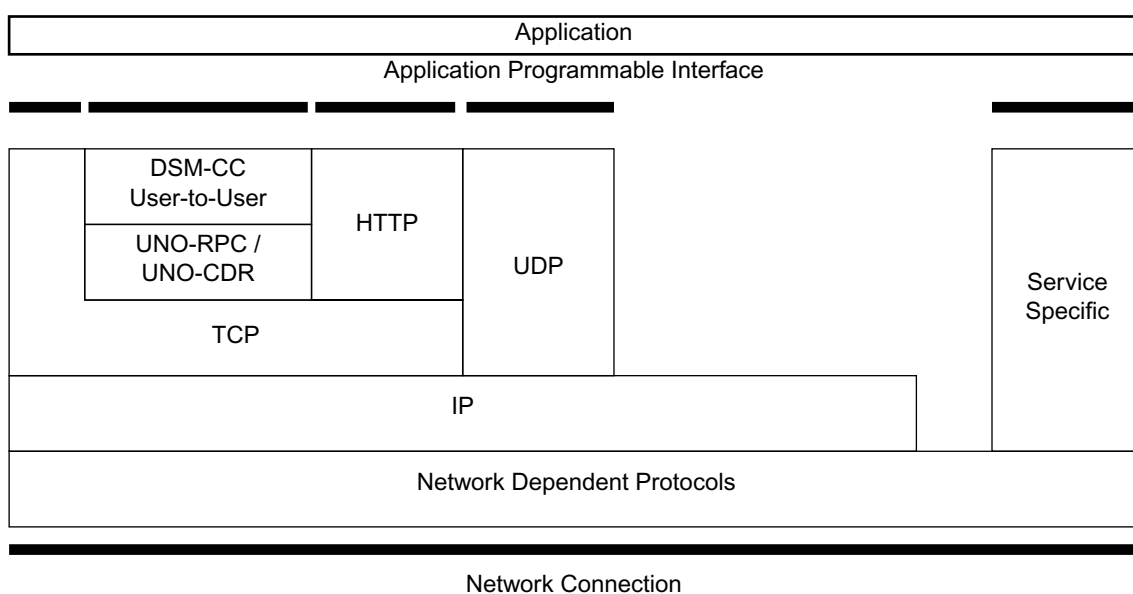


Figure 9 : Interaction Channel Protocol Stack

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 \[48\]](#), [EN 301 790 \[83\]](#) respectively for CATV, PSTN/ISDN, DECT, GSM, LMDS SMATV and satellite 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 \[40\]](#).

6.3.8 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 \[42\]](#).

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 53.

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 218.

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"

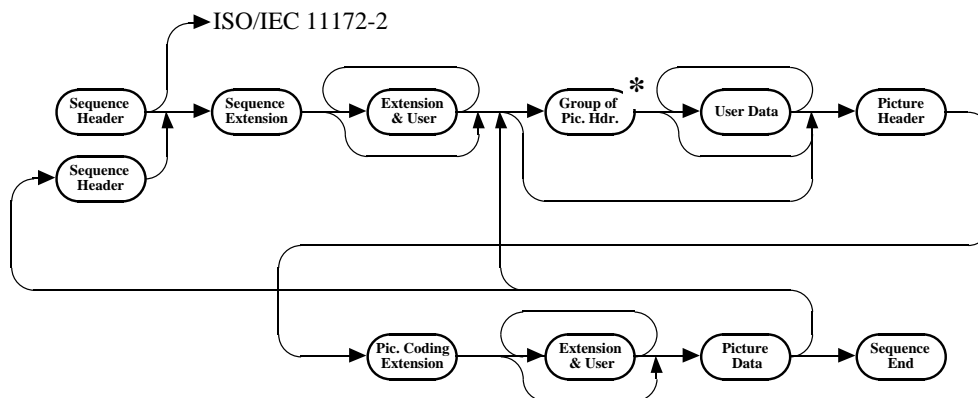
The drip feed mode consists of letting an application progressively feed the MPEG-2 video decoder with chunks of an MPEG-2 video stream. In this mode, it is only required for the decoder to handle I and P frames (i.e. not B frame). Each chunk shall contain one frame and a certain number of syntactic 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 465](#).

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 320](#).

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 204](#).

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 329.

See also annex D, "(normative): Text presentation" on page 302.

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 302.

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 1. 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 1 : 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 ²

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 2, 3. From these primaries the $Y C_b C_r$ transmitted values are computed by similar relationships.

Therefore ITU-R BT.709 $Y C_b C_r$ colourspace 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 2 : ITU-R BT.709 reference primaries and CIE standard illuminant

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

Table 3 : TU-R BT.470-2 reference primaries and CIE standard illuminant

	Red	Green	Blue	D65
x	0,6700	0,2100	0,1400	0,3100
y	0,3300	0,7100	0,0800	0,3160
z	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 4 : File type identification

MIME type	Extension (note 1)	Definition of content
"image/jpeg"	".jpg"	As defined in 7.1.1.2, "JPEG" on page 50.
"image/png"	".png"	As defined in 7.1.1.3, "PNG" on page 50 possibly with a constrained profile as defined in 15.1, "PNG - restrictions" on page 218.
"image/gif"	".gif"	As defined in 7.1.1.4, "GIF" on page 50.
"image/mpeg"	".mpg"	As defined in 7.1.2, "MPEG-2 I-Frames" on page 50.
"video/mpeg"	".mpg"	As defined in 7.2.2, "Video" on page 52.
"video/dvb.mpeg.drip"	".drip"	As defined in 7.1.3, "MPEG-2 Video "drips"" on page 50.
"audio/mpeg"	".mp2"	As defined in 7.1.4, "Monomedia format for audio clips" on page 52 or as defined in 7.2.1, "Audio" on page 52.
"text/dvb.utf8"	".txt"	As defined in 7.1.5, "Monomedia format for text" on page 52.
"image/dvb.subtitle"	".sub"	As defined in 7.2.3, "Subtitles" on page 52.
"text/dvb.subtitle"		
"text/dvb.teletext"	".tlx"	As defined in 7.2.3.2, "Teletext" on page 53.
"application/dvb.pfr"	".pfr"	As defined in 7.4, "Downloadable Fonts" on page 53.
"application/dvbj"	".class"	A DVB-J class file. See 6.2.5.1, "DVB-J class files" on page 46.
"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 41, "Return types of URL.getContent()" on page 107).

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 218. 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 Status of DVB HTML

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 67.
- How to signal a DVB HTML application in 10, "Application Signalling" on page 75.
- Extensions on how to transport a DVB HTML application in 6.2.5.2, "DVB-HTML document files" on page 47.

A definition of the content and application format elements from the HTML family is not in this release of the specification. Such a definition will be based on content formats defined by recognized bodies including W3C plus own developments.

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. Host Control tune requests from a CI module cause service selections. Host Control replace / clear_replace has an equivalent effect to using `javax.tv.media.MediaSelectControl`.

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.

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.

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.

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 81 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. Since the resident navigator is not required to provide a mechanism for this explicit launching, broadcast services wishing to have applications started must provide an application which is identified as auto-start.

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.

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 (e.g. memory, CPU and resources managed by the resource management API) to continue the execution of one of the running applications, the MHP terminal is allowed to decide to stop an MHP application without user intervention.

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 will not automatically gain access to any new broadcast file system on the new service. It remains logically attached to any broadcast file systems it has already attached to, although it may be temporarily disconnected from the broadcast carousels feeding those file systems. The behaviour of the broadcast file system under these circumstances is defined in 6.2.5.3, "Loss of Carousel Behaviour" on page 47.

If temporarily disconnected, the broadcast carousel shall be reconnected upon selection of a service where the carousel is available. This includes, but is not exclusively, the original service the application ran in. To determine carousel availability, the MHP shall use the PMT information obtained for the "home" service of the carousel at the time of mounting, and the PMT of the currently selected service.

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 itself, it shall not be 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.

In this case the platform may attempt to re-start the application however this is implementation dependent.

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 78. 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

9.1.8 DVB-J Applications and Service Selection

DVB-J 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.

DVB-J 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`.

DVB-J applications may cause tuning to another transport stream using the tuning API. Usage of these APIs does not constitute service selection and therefore no applications from the target transport stream or service of these APIs 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 78. The service being presented in the service context shall not be changed by usage of these APIs.

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 608 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 initial class 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 11 shows the state machine model for Xlets. The Xlet states are defined in more detail in table 5.

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 77)).
- 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 127).

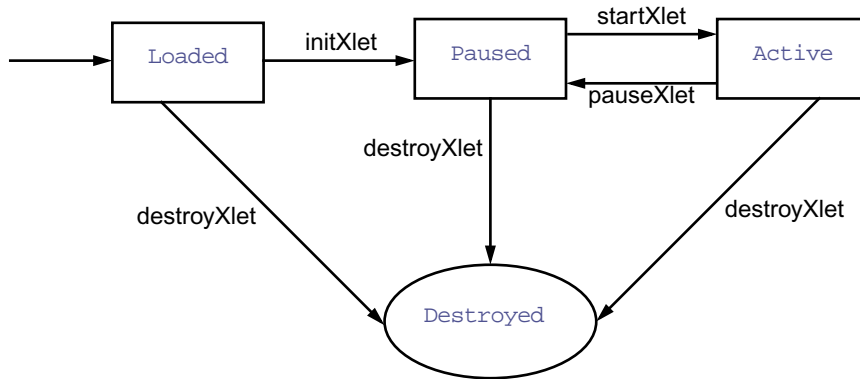


Figure 11 : Xlet lifecycle state machine diagram

Table 5 : 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) shall immediately enter the <code>Destroyed</code> state. 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>Notes:</p> <ul style="list-style-type: none"> • Application initialisation is intended to occur in the <code>initXlet</code> method, rather than in the default constructor. • This state is entered only once per instance of an DVB-J application.
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"> • 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. <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"> • 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. • From the <code>Active</code> state upon entering the <code>XletContext.notifyPaused()</code> method.

Table 5 : 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"> From the <code>Paused</code> state after the <code>Xlet.startXlet()</code> method returns successfully.
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"> 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 the Xlet may be garbage collected. Upon entering the <code>XletContext.notifyDestroyed()</code> method. The Xlet performs its clean up actions before calling the <code>notifyDestroyed()</code> method. <p>NOTE: This state is entered only once per instance of an DVB-J application.</p>

Every time a DVB-J application instance is started (i.e. the constructor of the object implementing Xlet is called), it shall logically run in its own new virtual machine instance. See 11.2.1, "Basic Considerations" on page 102.

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 6 : 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 126.

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 the previous states in which calls to the methods on `XletContext` relating to state management are valid;

NOTE: These methods are called after the Xlet has changed its state.

Table 7 : States for valid state management calls

Call	State
<code>notifyDestroyed</code>	all states
<code>notifyPaused</code>	active 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 127](#).

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 108](#).

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 131 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 102.

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.

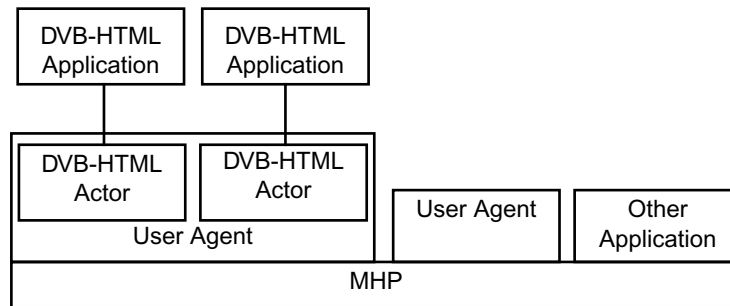


Figure 12 : Relationships between actors and applications

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 determine when one DVB-HTML application should start another, 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 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 99](#)). 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 75.

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 81) 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.

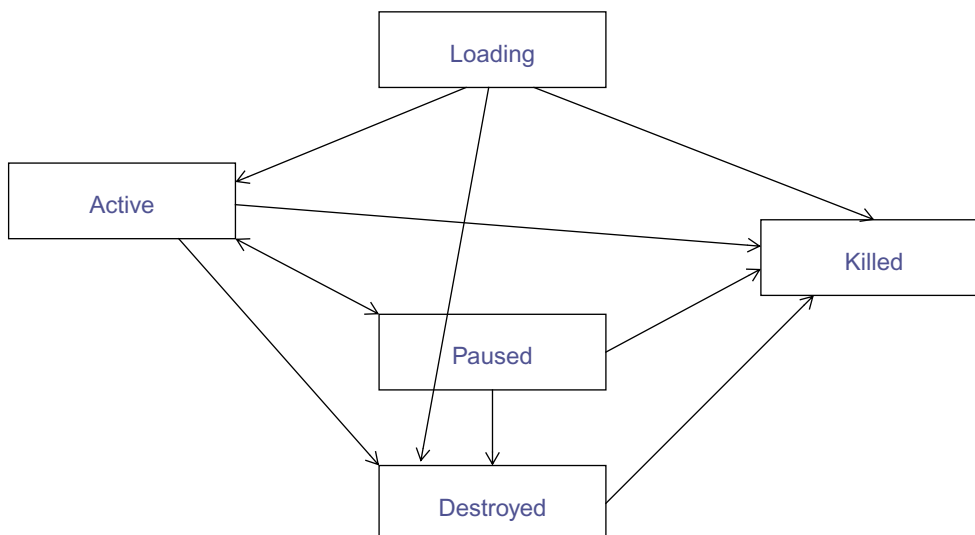


Figure 13 : DVB-HTML application life cycle state diagram

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 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. When comparing two applications, the one with the higher application_priority value shall be considered the more important to preserve.

- 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 or the external application authorisation descriptor depending on which descriptor the application is currently signalled by.
- Unless stated otherwise, only the application which owns a resource has the right to modify the state / settings / configuration of that resource. If a resident application (e.g. the MHP navigator) makes changes to the state / settings / configuration of a resource, the MHP terminal shall inform the MHP application which formerly owned the resource that the MHP application has lost ownership of the resource.

NOTE: The only exception to this rule is where all applications running inside a service context together own the service component handlers of that service context and not any single one of them. See 11.6.2, "Service Selection API" on page 123.

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 27

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 28](#).

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 26, "Semantic of selector bytes" on page 90](#)
- 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 95](#)
- Define the application type specific life cycle control codes in [10.6, "Control of application life cycle" on page 81](#).

Where constant values are registered by this specification extend the [table 39, "Registry of constant values" on page 100](#).

10.1.7 Service information

See [10.12, "Service Information" on page 101](#).

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 77](#)).

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.2.10, "Mounting an Object Carousel" on page 292](#).

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.2.10, "Mounting an Object Carousel" on page 292](#)).

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 84](#).

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 a descriptor shall result in that descriptor being silently discarded. Processing of that descriptor loop shall continue with the next descriptor (if any). The scope of error detection of a descriptor should be limited to the application information section in which it is carried.
- An error in an application loop outside a descriptor 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).

NOTE: The consequence of the above is that an error in a mandatory descriptor which results in that descriptor being silently ignored may then result in a application loop which is missing such a mandatory descriptor. Hence that application loop shall also be silently ignored.

- 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 application types a terminal can decode include those types for which the terminal has available interoperable plug-ins (see figure 7, "Illustrative plug-in implementation options" on page 44).

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 83.

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 8 : 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

Table 8 : Application Information Section syntax (Sheet 2 of 2)

	No. of Bits	Identifier
<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<N;i++){</code> <code> descriptor()</code> <code>}</code>		
<code>reserved_future_use</code>	4	bslbf
<code>application_loop_length</code>	12	uimsbf
<code>for(i=0;i<N;i++){</code> <code> application_identifier()</code> <code> application_control_code</code>	8	uimsbf
<code> reserved_future_use</code>	4	bslbf
<code> application_descriptors_loop_length</code>	12	uimsbf
<code> for(j=0;j<N;j++){</code> <code> descriptor()</code> <code> }</code> <code>}</code>		
<code>CRC_32</code>	32	rpchof
<code>}</code>		

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 9.

Table 9 : Application types

application_type	description
0x0000	reserved_future_use
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".

current_next_indicator: This 1-bit indicator shall be set to "1".

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 81.

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 80.

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 52). See also 14.5, "Text encoding of application identifiers" on page 212.

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 10 : 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 TR 101 162 [10]. Values of zero shall not be encoded.

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 155).

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. Values of zero shall not be encoded.

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 145). 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 11 : 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 89. 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 608.

10.5.3 Authentication of application identification

See 12.5.6, "subject" on page 155.

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 58.

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 `AppStateChangeEvent` 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 12.

Table 12 : DVB-J application control code values

code	identifier	semantics
0x00		reserved_future_use
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 <code>unconditional</code> parameter set to <code>false</code>) 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 <code>unconditional</code> parameter set to <code>true</code>) by the application manager.
0x05		reserved_future_use
0x06	REMOTE	This identifies a remote application that is only launchable after service selection.
0x07...0xFF		reserved_future_use

See 9.2.3, "DVB-J Application Lifecycle" on page 62.

10.6.2.2 DVB-HTML

The application control codes for DVB-HTML applications are listed in 13.

Table 13 : DVB-HTML application control code values

code	identifier	semantics
0x00		reserved_future_use
0x01	AUTOSTART	The Application Entry Point of the DVB-HTML application is loaded. This is loaded into the user agent, and the DVB-HTML actor is created (in the Loading state) and the DVB-HTML application is started. When these steps are complete the DVB-HTML actor is in the Active 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 Killed 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 Active state and waits for a trigger before completely transitioning to the Active state.
0x06	REMOTE	This identifies a remote application that is only launchable after service selection.
0x07...0xFF		reserved_future_use

See 9.3.2, "[DVB-HTML Application Lifecycle](#)" on page 69.

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 78.

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 77.

Table 14 : 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<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 15). Where no "id specific data" is provided the descriptor just identifies the "principal" component of a data broadcast.

Table 15 : 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	
<code>for (i=0; i<N; i++) {</code>			
<code>id specific data</code>	8	bslbf	
<code>}</code>			
<code>}</code>			

10.7.2.2 MHP data broadcast id descriptor

When the data broadcast id is one of those defined by this specification (see table 39) the syntax of the data broadcast id descriptor is as shown in table 16. This extends the generic descriptor with an optional list of application types for which autostart applications may exist within the data broadcast. This list provides a hint to allow the MHP terminal to prioritise connection to a data broadcast when several are provided by the service. If no list is provided then the data broadcast id descriptor is silent on the types of autostart applications that may be carried by the data broadcast. If the application list is not empty, then the data broadcast shall not include autostart applications of application types other than those in the list. It is not required that the data broadcast always include autostart applications of all types in the list.

Table 16 : 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	
<code>for (i=0; i<N; i++) {</code>			
<code>application_type</code>	16	uimsbf	
<code>}</code>			
<code>}</code>			

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 9 on page 79.

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 17 : application descriptor syntax

	No.of Bits	Identifier	Value
<code>application_descriptor() {</code>			
<code>descriptor_tag</code>	8	uimsbf	
<code>descriptor_length</code>	8	uimsbf	
<code>application_profiles_length</code>	8	uimsbf	
<code>for(i=0; i<N; i++) {</code>			
<code>application_profile</code>	16	uimsbf	
<code>version.major</code>	8	uimsbf	
<code>version.minor</code>	8	uimsbf	
<code>version.micro</code>	8	uimsbf	
<code>}</code>			
<code>service_bound_flag</code>	1	bslbf	
<code>visibility</code>	2	bslbf	
<code>reserved_future_use</code>	5	bslbf	
<code>application_priority</code>	8	uimsbf	
<code>for(i=0; i<N; i++) {</code>			
<code>transport_protocol_label</code>	8	uimsbf	
<code>}</code>			
<code>}</code>			

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:

$$\begin{aligned} & (\text{application_profile} \in \text{terminal_profiles_set}) \\ & \wedge \{ (\text{application_version.major} < \text{terminal_version.major}(\text{application_profile})) \\ & \vee [(\text{application_version.major} = \text{terminal_version.major}(\text{application_profile})) \\ & \wedge (\{ \text{application_version.minor} < \text{terminal_version.minor}(\text{application_profile}) \} \\ & \vee \{ [\text{application_version.minor} = \text{terminal_version.minor}(\text{application_profile})] \\ & \wedge [\text{application_version.micro} \leq \text{terminal_version.micro}(\text{application_profile})] \})] \} \end{aligned}$$

Where:

\in represents "belongs to the set of"
 \wedge represents "logical AND"
 \vee represents "logical OR"

See table 69, "Profile encoding" on page 220 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 whether the application is suitable to be offered to the end-user for them to decide if the application should be launched. Table 18 lists the allowed values of this field.

NOTE: This applies equally to any generic launching menu application provided in the MHP service and to any application launching user interface provided in the MHP navigator.

Table 18 : 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_future_use
11	This application can be visible to users and shall be visible to applications via the application listing API.

NOTE: For example, in a service offering a number of games to the end-user, these values would be used as follows:

00 - the autostart generic launcher application offering the end user the choice of which game to launch

11 - the games which the end-user can chose between

01 - the common server application which all the games use to communicate with a server over the interaction channel.

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 19 : application name descriptor syntax

	No.of Bits	Identifier	Value
<code>application_name_descriptor() {</code>			
<code>descriptor_tag</code>	8	uimsbf	
<code>descriptor_length</code>	8	uimsbf	
<code>for (i=0; i<N; i++) {</code>			
<code>ISO_639_language_code</code>	24	bslbf	
<code>application_name_length</code>	8	uimsbf	
<code>for (i=0; i<N; i++) {</code>			
<code>application_name_char</code>	8	uimsbf	
<code>}</code>			
<code>}</code>			
<code>}</code>			

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 8859-1 \[20\]](#) 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 218.

Table 20 : application icons descriptor syntax

	No.of Bits	Identifier	Value
<code>application_icons_descriptor() {</code>			
<code>descriptor_tag</code>	8	uimsbf	
<code>descriptor_length</code>	8	uimsbf	
<code>icon_locator_length</code>	8	uimsbf	
<code>for (i=0; i<N; i++) {</code>			
<code>icon_locator_byte</code>	8	uimsbf	
<code>}</code>			
<code>icon_flags</code>	16	bslbf	
<code>for (i=0; i<N; i++) {</code>			
<code>reserved_future_use</code>	8	bslbf	
<code>}</code>			
<code>}</code>			

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 21. The `icon_locator` shall not end with a "/" slash character.

Table 21 : Icon locator semantics

application_type	description
0x0000	<code>reserved_future_use</code>
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 96.
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 97.
0x0003...0xFFFF	<code>reserved_future_use</code>

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 22.

Table 22 : Definition of different icon flags (Sheet 1 of 2)

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

Table 22 : Definition of different icon flags (Sheet 2 of 2)

Icon flag bits	Description of icon size and pixel aspect ratio
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_future_use
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"
```

An icon file shall contain exactly one icon. The icon contained in the icon file shall have the format specified by the 4 hexadecimal digit postscript of its file name.

NOTE: This means that if the `icon_flags` field of the `application_icons_descriptor` were to have a value indicating the presence of multiple icons, each of the indicated icons would have its own icon file. For example, if `icon_flags` has a value of 0x0005, the directory specified by `icon_locator` would contain two files named `dvb.icon.0004` (for 64x64 square pixel rendering) and `dvb.icon.0001` (for 32x32 square pixel rendering).

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 23 : external application authorisation descriptor syntax

	No. of Bits	Identifier	Value
<code>external_application_authorisation_descriptor() {</code>			
<code>descriptor_tag</code>	8	uimsbf	
<code>descriptor_length</code>	8	uimsbf	
<code>for(i=0; i<N; i++) {</code>			
<code>application_identifier()</code>			
<code>application_priority</code>	8	uimsbf	
<code>}</code>			
<code>}</code>			

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 80.

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 85).

See `application_priority` under 10.7.3, "Application descriptor" on page 85.

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 24 : transport protocol descriptor syntax

	No.of Bits	Identifier	Value
<code>transport_protocol_descriptor() {</code>			
<code>descriptor_tag</code>	8	uimsbf	
<code>descriptor_length</code>	8	uimsbf	
<code>protocol_id</code>	16	uimsbf	
<code>transport_protocol_label</code>	8	uimsbf	
<code>for(i=0; i<N; i++) {</code>			
<code>selector_byte</code>	8	uimsbf	N1
<code>}</code>			
<code>}</code>			

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 25 : Protocol_id

protocol_id	description
0x0000	reserved_future_use
0x0001	MHP Object Carousel as defined in annex B, "(normative): Object carousel" on page 267.
0x0002	IP via DVB multiprotocol encapsulation as defined in EN 301 192 [5], TR 101 202 [49]
0x0003...0x00FF	reserved_future_use
0x0100...0xFFFF	Subject to registration in TR 101 162 [10]

transport_protocol_label: This 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 26 : Semantic of selector bytes

protocol_id	selector byte data
0x0000	reserved_future_use
0x0001	See 10.8.1.1 "Transport via OC".
0x0002	See 10.8.1.2 "Transport via IP".
0x0003...0xFFFF	Not defined in this version of the specification

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 27.

Table 27 : 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. 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). When this bit is 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.

Applications with this flag set shall have their application control code set to REMOTE (see table 12 on page 82 and 13 on page 83).

See 11.7.2, "Application discovery and launching APIs" on page 127.

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 28.

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 216.

Table 28 : Syntax of selector bytes for IP transport (Sheet 1 of 2)

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

Table 28 : Syntax of selector bytes for IP transport (Sheet 2 of 2)

Syntax	Bits	Mnemonic
for(i=0; i<N; i++){ URL_length	8	uimsbf
for(j=0; j<URL_length; j++){ URL_byte	8	uimsbf
} }		

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 91.

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].

10.8.2 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 29 : Syntax of the IPv4 routing descriptor

Syntax	Bits	Mnemonic
routing_descriptor_ip4 () {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for(i=0; i<N; i++) {		
component_tag	8	uimsbf
address	32	uimsbf
port_number	16	uimsbf
address_mask	32	uimsbf
}		
}		

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 30 : 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<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.

For one application, the pre-fetch descriptor(s) and possible DII location descriptor present in the AIT shall point to the same transport connection. If the `transport_protocol_labels` present in these descriptors are different, the referenced transport protocol descriptor shall point to the same transport connection (carousel).

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 269).

Table 31 : 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<N; i++) {</code>		
<code>label_length</code>		
<code>for(j=0; j<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 object carousel that delivers the modules to which this prefetch descriptor refers. See "[transport_protocol_label](#)" on page 87.

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 269).

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 94), 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 32 : Syntax of the DII location descriptor

Syntax	Bits	Mnemonic
<pre>DII_location_descriptor () { descriptor_tag descriptor_length transport_protocol_label for(i=0; i<N; i++) { reserved_future_use DII_identification association_tag } }</pre>	<p>8</p> <p>8</p> <p>8</p> <p>1</p> <p>15</p> <p>16</p>	<p>uimsbf</p> <p>uimsbf</p> <p>uimsbf</p> <p>bslbf</p> <p>uimsbf</p> <p>uimsbf</p>

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 object carousel that delivers the modules to which this prefetch descriptor refers. See "[transport_protocol_label](#)" on page 87.

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 290.

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 33 : DVB-J application descriptor syntax

	No. of Bits	Identifier	Value
<pre>dvb_j_application_descriptor(){ descriptor_tag descriptor_length for(i=0; i<N; i++) { parameter_length for(j=0; j<parameter_length; j++) { parameter_byte } } }</pre>	<p>8</p> <p>8</p> <p>8</p> <p>8</p>	<p>uimsbf</p> <p>uimsbf</p> <p>uimsbf</p> <p>uimsbf</p>	

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 34 : DVB-J application location descriptor syntax

	No. of Bits	Identifier	Value
dvb_j_application_location_descriptor {			
descriptor_tag	8	uimsbf	
descriptor_length	8	uimsbf	
base_directory_length	8	uimsbf	
for(i=0; i<N; i++) {			
base_directory_byte	8	uimsbf	
}			
classpath_extension_length	8	uimsbf	
for(i=0; i<N; i++) {			
classpath_extension_byte	8	uimsbf	
}			
for(i=0; i<N; i++) {			
initial_class_byte	8	uimsbf	
}			
}			

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.

The value of this field shall be at least one.

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"). The length of this string must be at least one.

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 35 : DVB-HTML application descriptor syntax

	No. of Bits	Identifier	Value
<code>dvb_html_application_descriptor(){</code>			
<code>descriptor_tag</code>	8	uimsbf	
<code>descriptor_length</code>	8	uimsbf	
<code>appid_set_length</code>	8	uimsbf	N1
<code>for(i=0; i<N1; i++) {</code>			
<code>application_id</code>	16	bslbf	
<code>}</code>			
<code>for(j=0; j<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 80). The semantics of this information is not defined in this specification.

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 36 : DVB-HTML application location descriptor syntax

	No. of Bits	Identifier	Value
<code>dvb_html_application_location_descriptor () {</code>			
<code>descriptor_tag</code>	8	uimsbf	
<code>descriptor_length</code>	8	uimsbf	
<code>physical_root_length</code>	8	uimsbf	N1
<code>for(i=0; i<N1; i++) {</code>			
<code>physical_root_bytes</code>	8	uimsbf	
<code>}</code>			
<code>for(i=0; i<N; i++) {</code>			
<code>initial_path_bytes</code>	8	uimsbf	
<code>}</code>			
<code>}</code>			

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 37 "Transport specific semantic of physical root bytes".

Table 37 : Transport specific semantic of physical root bytes

protocol_id	semantic
0x0000	reserved_future_use
0x0001	A directory specification
0x0002	One of the base URLs defined in the Transport protocol descriptor signalled for the application (see 10.8.1.2, "Transport via IP" on page 91).
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 38 : 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<N1; i++) { label_bytes } for(i=0; i<N; i++) { regular_expression_bytes } } </pre>	<p>8</p> <p>8</p> <p>8</p> <p>8</p> <p>8</p>	<p>uimsbf</p> <p>uimsbf</p> <p>uimsbf</p> <p>uimsbf</p> <p>uimsbf</p>	<p>N1</p>

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 68.

10.11 Constant values

Table 39 : Registry of constant values

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					
reserved to MHP for future use		0x0E-0x5E					
private data specifier descriptor (note 2)		0x5F					
reserved to MHP for future use		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 for future use	0x00F0 - 0x00FE		PMT, AIT	SI			
MHP Application Service	service type	0x10	SDT	SI			
<p>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.</p> <p>NOTE 2: The DVB SI private data specifier descriptor is defined for use in the Application Information Table to introduce private descriptors.</p> <p>NOTE 3: All user defined descriptors shall be within the scope of a private data specifier descriptor (see 10.4.7, "Use of private descriptors in the AIT" on page 80).</p>							

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 214.

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 40 : 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.

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. Where a class is defined by this specification as implementing a specific interface, and that interface requires the class to provide an otherwise deprecated method. Then the interface overrides the deprecated mark and the method is required by this specification.

Each DVB-J application instance 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 VM \[34\]](#).

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.

Applications shall not define classes or interfaces in any package namespace defined in this specification. MHP terminals shall enforce this using the `SecurityManager.checkPackageDefinition` mechanism.

DVB-J applications that have been written to be scalable across multiple MHP profiles and/or versions of profiles and/or optional features may include references to API classes that are only present if the MHP terminal implements the profile or version of a profile or an optional feature that these API classes represent. MHP terminals shall ensure that any error conditions due to inability to link classes in DVB-J applications are thrown only if the execution of the application reaches a bytecode where these classes are referenced from the application. The presence of the reference in the class file to a non-existing class shall not lead to an error being thrown when loading the referencing class.

NOTE: As long as the application properly tests for the presence of these API features prior to calling them in a conditional branch, the application shall be able to execute on MHP terminals that do not have these API features.

11.2.2 Approach to Subsetting

Where a class included in this specification has methods, fields or constructors 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.

In a signed application, all classes or files to be loaded through the classpath shall be signed by at least the set of certificates used to sign the initial xlet class of the application. This applies, for example, to class files comprising the application, and images and other data loaded via the `java.lang.Class.getResource()` mechanism. When authenticating a signed application, an MHP terminal can select any one of these certificates to use to authenticate all subsequent classes or files loaded. The mechanism used to make this selection is implementation dependent.

NOTE: Where an MHP terminal trusts more than one of the certificates used to sign the initial xlet, it should attempt to select the most trusted of these.

See 12, "Security" on page 145.

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 DAVIC APIs [DAVIC 1.4.1p9 \[3\]](#) which currently directly extend `java.lang.Object` 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`, unless explicitly specified otherwise, 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 `java.lang.Thread.NORM_PRIORITY`.

NOTE: As a consequence applications will not be able to create threads at priorities greater than `java.lang.Thread.NORM_PRIORITY` since they don't have `java.lang.RuntimePermission("modifyThread")`. Applications may perform compute intensive tasks within application created threads without being considered unresponsive.

11.2.11 Text Encodings

Where the specification of the Java APIs refers to the default character encoding of the platform, the default for MHP shall be "UTF8" as defined in 7.1.5, "Monomedia format for text" on page 52. The encoding "latin1" shall also be supported as defined in ISO 8859-1 [20].

When present in a Java `String`, the mark-up codes defined in table D.7, "Codes defined for use in marked-up text files" on page 318 shall be encoded in Java chars whose most significant byte is zero and whose least significant byte is the value from table D.7 on page 318. The encoding "DVBMarkupUTF8" shall be supported and is defined to be the same as UTF8 except as follows:

- In byte to char translations, the mark-up sequences in table D.7 on page 318 shall be translated into chars as defined above.
- In char to byte translations, sequences of characters matching the encodings above shall be translated into the corresponding mark-up code sequences in table D.7 on page 318.

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 133.

11.3.1.1 java.lang package

The `java.lang` package is supported with the following modifications.

- 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 122)

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 \[36\]](#).

- j) The in [JDK 1.2.2 \[79\]](#) the methods `java.lang.SecurityManager.checkPackageDefinition` and `java.lang.SecurityManager.checkPackageAccess` are documented to check a package name against a list of packages obtained from a method defined in `java.security.Security`. As the class `java.security.Security` is not required by MHP, for MHP the documentation of these two methods is considered to say that the list of restricted packages is obtained from a platform-specific list.

MHP terminal implementors are recommended to use this mechanism to prevent application classes from having access to implementation classes where this could compromise the security of the MHP terminal.

- NOTE: MHP terminal implementors who are also implementing MHP applications need to ensure that the namespaces used for applications do not collide with those used for their MHP implementation.

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 219).

The format used for the `java.util.Properties.save()` and `java.util.Properties.load()` methods shall be that specified for those methods in [JDK 1.2.2 \[79\]](#).

The effect of an MHP application calling the `java.util.Timezone.setDefault` method shall be limited to the application calling the method.

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.

The method `java.io.ObjectInputStream.readLine()` shall not be called by inter-operable applications but is required to be present for backwards compatibility with earlier versions of the MHP specification.

The classes and interfaces in this package relating to files and file systems have additional semantics defined for MHP specific file systems as follows.

- For broadcast carousels in 11.5.1.1, "Constraints on the `java.io.File` methods for broadcast carousels" on page 119
- For persistent storage in 11.5.6, "Persistent Storage API" on page 122

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 103. The signature dependencies from these to the rest of this package are severed as described in 11.2.2, "Approach to Subsetting" on page 103.

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.

In a signed application, a URL to this resource shall be returned as for an unsigned application, regardless of whether the underlying file is signed. When that file is accessed (e.g. via an input stream obtained from `java.net.URL.openStream()`), then if the file fails authentication as described in 11.2.3, "Class Loading" on page 103, the system shall behave as if the file contained no data (i.e. as if it were a zero-length file).

NOTE: Due to the overhead of processing the signature verification, asset files which are not critical to be authenticated should be loaded using `java.io.File` or `org.dvb.dsmcc.DSMCCObject` and sent as unsigned.

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 41 "Return types of `URL.getContent()`".

Table 41 : 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 273).
 - the HTTP header (if supported in the profile) the Content-Type header field (if present)
- b) The filename extension (if known) (see table 4, "File type identification" on page 56)
- 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 273) if present.

`getFileNameMap` returns information derived from table 4, "File type identification" on page 56.

In the class `java.net.URLStreamHandler`, the protected method `setURL` is not required to be present.

In the class `java.net.URLConnection`, the static methods `getDefaultRequestProperty` and `setDefaultRequestProperties` are not required to be present.

See the Object Carousel signalling described in B.2.3.4, "Content type descriptor" on page 273.

11.3.1.7 java.beans

In this package the following classes and interfaces are supported;

- Beans
- PropertyChangeEvent
- PropertyChangeListener
- PropertyChangeSupport
- PropertyVetoException
- VetoableChangeListener
- VetoableChangeSupport
- PropertyVetoException
- Visibility

In the Beans class, only the following methods shall be required:

- Beans.instantiate(ClassLoader cl, String beanName)
- Beans.isDesignTime()
- Beans.isGUIAvailable()

Finally, the following methods shall behave as follows;

- Beans.isDesignTime() must return "false".

11.3.1.8 java.math

From the java.math package java.math.BigInteger is supported.

11.3.2 MHP platform APIs

11.3.2.1 org.dvb.lang

The org.dvb.lang package is supported as defined in annex I, "(normative): DVB-J fundamental classes" on page 334.

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 J, "(normative): DVB-J event API" on page 338.

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`
- `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 133](#).

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 103](#). The same applies for all references to the `java.awt.peer` package. In `Component` and `Container`, the `list` methods are not required.

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 \[50\]](#).

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 50. 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 302](#)) 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 653](#) 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.

Applications shall only pass in calls to the method `javax.tv.graphics.TVContainer.getRootContainer` the exact same `XletContext` instance as was passed to their `initXlet` method when it was called by the MHP terminal. The behaviour of `getRootContainer` if passed any other `XletContext` is implementation dependent and returning null is one valid option. Except for this case, this method shall never return null in an MHP implementation.

The method `HComponent.isDoubleBuffered` shall not be used by inter-operable applications.

11.4.1.3 Extended graphics

See [U, "\(normative\): Extended graphics APIs" on page 653](#).

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 108](#)). 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 330](#).

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 an `HScene` 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 [J.1 on page 338](#).
- 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` of the `commit<xxx>` methods shall always be thrown in MHP.

The events `javax.media.ControllerEvent` and `javax.media.GainChangeEvent` shall inherit from `java.util.EventObject`.

The interfaces `javax.media.ControllerListener` and `javax.media.GainChangeListener` shall inherit from `java.util.EventListener`.

`SubtitlingLanguageControl.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 50):

- `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 465 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.

NOTE: This specification defines more specific conditions under which `ResourceWithdrawnEvent` and `ResourceReturnedEvent` than the original DAVIC specification. This can be found in 11.6.2, "Service Selection API" on page 123.

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.

The components returned by `javax.tv.media.MediaSelectControl.getCurrentSelection` are the currently selected components which may be different from those previously selected by this control. Mechanisms which may modify the current selection include:

- control of subtitles & audio language
- the CA system including the common interface e.g Host Control replace / clear_replace
- user intervention via the navigator

The following clarifications shall apply to the method `getDefaultSize` in `javax.tv.media.AWTVideoSizeControl`:

- a) This method shall return the current default `AWTVideoSize`. The default shall be that which would be implemented by the MHP terminal when the video scaling & positioning of a JMF player is under control of the MHP platform (see `org.dvb.media.VideoFormatControl.DFC_PLATFORM`). When the video scaling and positioning is in the `DFC_PLATFORM` mode, the return value of this method shall change to track changes made by the policy underlying `DFC_PLATFORM`.
- b) Calling `AWTVideoSizeControl.setSize()` with the `AWTVideoSize` object returned by this method as the parameter shall set the video scaling to the current default at the time `getDefaultSize()` was called. Hence it shall be equivalent to calling `BackgroundVideoPresentationControl.setVideoTransformation` with the video transformation returned by the method `VideoFormatControl.getVideoTransformation(getDecoderFormatConversion())` and not with the video transformation returned by `getVideoTransformation(DFC_PLATFORM)`.

NOTE: When the MHP terminal is in pan & scan mode, (see `org.dvb.media.VideoFormatControl.DFC_PROCESSING_PAN_SCAN`), the return value of `AWTVideoSizeControl.getSize()` will be out of date almost as soon as the method has returned.

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 52:

- `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`
- `org.dvb.media.DVBMediaSelectControl`

The following controls are supported for media decoded from 7.1.4, "Monomedia format for audio clips" on page 52:

- `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 204.

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 193.

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.

NOTE: In this version of this specification, no media types are defined for which `javax.media.Player.addController` can have any other behaviour apart from throwing a `IncompatibleTimeBaseException`.

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 34, "Determining subtitling language and presentation setting" on page 204 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 17 on page 189. 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 31, "Format control in the presence of a JMF player" on page 202, 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 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 an 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 517.

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 96. 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 96.

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 `DSMCCObject` 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 `FileInputStream`, `FileReader` or `RandomAccessFile` is called will not be visible in the data read from that instance.
- list methods shall return null when called on a object that isn't a directory.
- the creation of a `FileInputStream`, `FileReader` or `RandomAccessFile` shall throw `FileNotFoundException` if the referenced object is not a file.
- Failure of a signed file to be authenticated shall be reported as defined in 12.6.1, "General principles" on page 157.
- 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 carousel

When a broadcast carousel is lost to an application as described in 6.2.5.3, "Loss of Carousel Behaviour" on page 47, 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 119.
- 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.
- Operations ongoing at the time of loss of broadcast carousel shall be terminated. Blocked Java I/O operations shall throw `InterruptedIOException`.

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 582.

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.

The methods `getTTL` and `setTTL` of `java.net.MulticastSocket` are not required to be present.

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 135.

See also 11.8.1.3, "[Other classes](#)" on page 133.

The constant `java.net.HTTPURLConnection.HTTP_SERVER_ERROR` is not required to be present.

The class `java.net.DatagramSocketImpl` is not required to be present.

The methods `getTTL` and `setTTL` of `java.net.MulticastSocket` are not required to be present.

In the class `java.net.URLStreamHandler`, the protected method `setURL` is not required to be present.

In the class `java.net.URLConnection`, the static methods `getDefaultRequestProperty` and `setDefaultRequestProperties` are not required to be present.

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 585.

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 K, "(normative): DVB-J persistent storage API" on page 360.
- 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.
- Signed applications which are authenticated and which are granted the right to access persistent storage have the following privileges:
 - read only access to the root directory (defined above)
 - automatic read and write access to an "organisation" sub-directory named `<organisation_id>`
 - automatic read and write access to an "application" sub-directory named `<organisation_id> + <file.separator> + <application_id>`
- When the permission request file for an application requests access to persistent storage and this is granted, the MHP terminal shall be responsible for creating the necessary directories in which the application is allowed to read/write where these do not already exist. This shall be done before the first access (including existence checks such as `File.isDirectory()`) by that application instance to the directories concerned.

For applications which are granted file access (see 12.6.2.7, "File Access" on page 163), the necessary directories are only its organisation and application sub-directories. For applications which are granted a credential, (see 12.6.2.6, "Credentials" on page 161), the necessary directories are those defined by any filename elements in that credential up to but not including the first directory containing a wildcard.

EXAMPLE: A filename element such as

```
org0_id/appA_id/external/B_?/-
```

creates

```
org0_id/appA_id/external
```

When the MHP terminal automatically creates an "application" sub-directory as part of the necessary directories defined above, it shall set the owner of the created sub-directory to the application whose name corresponds to that sub-directory and set read and write access to the application whose name corresponds to that sub-directory and no access for other applications. The MHP terminal shall set the owner of any additional automatically created necessary directories that reside below an "application" sub-directory to the application whose name corresponds to that "application" sub-directory and set read and write access to the application whose name corresponds to that "application" sub-directory and no access for other applications.

If at the time when an MHP terminal would create the "application" sub-directory (if it did not exist), that directory already exists but is owned by an application other than the one whose application_id is used as its name, the MHP terminal shall remove this directory and any contents of it and create an empty "application" sub-directory with the owner and access rights set as defined above.

- The owner of the "organisation" directory shall be the platform and the directory itself shall always have organisation read / write and world read access.
- 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.
- See 14.5, "Text encoding of application identifiers" on page 212. 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 80.
- All access to persistent storage shall be consistent with the security mechanism defined in 12, "Security" on page 145.

NOTE 1: As defined in 11.5.1, "Broadcast Transport Protocol Access API" on page 118, relative paths cannot be used to access persistent storage in an inter-operable way. NOTE 2: Unsigned applications have no access to persistent storage. See 12.6.2.7.1, "Unsigned applications" on page 163

The contents of a file placed in persistent storage shall persist at least until one of the following conditions occur.

- a) There is not enough free persistent storage available to satisfy the persistent storage needs of a running application.
- b) The file stored in persistent storage expires.
- c) There is an interruption in power to the terminal.
- d) The file is cleared (i.e. deleted) by the creating application or an application with appropriate permissions.
- e) The file is cleared as a result of an explicit request by the end user.
- f) The persistent storage used by MHP applications exceeds 75% of the value in G.6, "Memory" on page 330

NOTE: This enables terminal's resource management system to release files as a result of an implementation-specific resource management policy.

The terminal shall give priority to the retention of higher priority files over lower priority files within the scope of a specific(organization ID, application ID) pair. See `org.dvb.io.persistent.FileAttributes` for priorities.

The details regarding the release of cleared (i.e.deleted) or expired files are implementation dependent.

The MHP terminal shall have a means to clear at least the quantity of persistent storage defined in G.6, "Memory" on page 330. This mechanism shall be usable in conformance tests, and is not subject to the rules above.

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 380.

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. The same JMF player shall be returned for all real-time media components sharing the same clock.

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 113.

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` methods where the new service to be selected is part of a transport stream known to the MHP terminal but not currently tuned to.

All the MHP applications running within the same service context shall have the ability to obtain and modify the state of the service component handlers for all service components being presented in that `ServiceContext`. For service component handlers which are JMF players, modifying the state includes changes to the settings of JMF controls in addition to the state machine of the JMF player itself.

Applications not running in that service context (e.g. the MHP navigator) shall not be able to modify the state of service component handlers unless they first inform the applications in the service context of this. For service component handlers which are JMF players, this informing shall be done by sending a `org.davic.media.ResourceWithdrawnEvent` to all applications which have currently registered listeners for `ControllerEvents` on the JMF player whose state will be modified.

Once a `ResourceWithdrawnEvent` has been sent, any changes made by applications inside the service context to the state of service component handlers will be ignored by the MHP terminal. Methods to query state shall return the actual state and methods to change the state shall silently fail.

If the application outside the service context ceases to need to make changes to the state of service component handlers, the applications inside the service context shall return to having that right exclusively. The applications inside the service context shall be informed of the return of this right. For service component handlers which are JMF players, this informing shall be done by sending a `org.davic.media.ResourceReturnedEvent` to all applications which have currently registered listeners for `ControllerEvents` on the JMF player which is returned to exclusive control.

Following receipt of a `ResourceReturnedEvent`, the MHP applications running in the service context are responsible for returning the state of all service component handlers to what they require. The MHP terminal is not required to automatically restore any state of any service component handler to any former value.

NOTE: One situation where this may occur is where the end-user of the MHP terminal brings up the UI of the navigator, does some operation and then exits the navigator, returning control back to the MHP applications.

NOTE: There is no relationship required between the generation of these events and the Xlet state model. MHP terminals may choose to put all MHP applications which are in the Active state into the Paused state when they bring up the UI of the navigator however there is no requirement for this.

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 133.

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 133.

Host Control tune requests from a CI module cause service selections. Host Control `replace / clear_replace` has an equivalent effect to using `javax.tv.media.MediaSelectControl`.

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 510.

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.

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 shall be supported:

- `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 95](#)) 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 80](#). Zero-length strings in the signalling shall be represented as the empty string.

The "dvb.caller.parameters" XletProperty contains the array of Strings that was passed into the `AppProxy.start(String[])` method by the application that started this application. If this application was started by the system or by another application using the `AppProxy.start()` method without parameters, an empty array with length 0 is returned as the value of this XletProperty.

Table 42 : Property name / type / encoding mapping

Property Name	<code>getXletProperty</code> return type
<code>dvb.app.id</code>	String encoded as in 14.5, "Text encoding of application identifiers" on page 212 .
<code>dvb.org.id</code>	String encoded as in 14.5, "Text encoding of application identifiers" on page 212 .
<code>dvb.caller.parameters</code>	String[], no information to be indicated with an array of length zero.

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 104](#).
- 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 608.

The following properties are defined for use with the method `AppAttributes.getProperty`:

Table 43 : Application attribute properties

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 String[] 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> .
<code>dvb.transport.oc.component.tag</code>	Returns Integer containing the <code>component_tag</code> from the selector bytes of the transport protocol descriptor. If more than one transport protocol descriptor is in the AIT for a remote application then it is implementation dependent which of them is returned.
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 version of this specification does not require support for applications signalled by any other means.

Table 44 : Information source for methods on AppAttributes

Method	Information source
<code>getName()</code>	One of the names that can be found in the application name descriptor.
<code>getName(String ISO639code)</code>	A name of the application from the application name descriptor corresponding to the specified language.
<code>getNames()</code>	All of the names for the application which can be found in the application name descriptor and their ISO 639 language code.
<code>getProfiles()</code>	The set of profiles signalled in the application profile field of the application descriptor.
<code>getPriority()</code>	The contents of the <code>application_priority</code> field of the application descriptor.
<code>getVersions(String profile)</code>	The values <code>version.major</code> , <code>version.minor</code> and <code>version.micro</code> for the specified profile from the application descriptor.
<code>getIsServiceBound()</code>	True if the <code>service_bound</code> field in the application descriptor is set to 1. Otherwise false.
<code>isStartable()</code>	There is no information source for this method, the return value is derived as specified in the method description. For the purpose of the method description, remote applications are defined to be those signalled as such in the transport protocol descriptor.
<code>getIdentifier()</code>	The contents of the <code>application_identifier</code> field of the application information section.
<code>getServiceLocator()</code>	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 45 : Information source for methods on Applcon

Method	Information source
getLocator()	The bytes carried in the <code>application_locator_byte</code> 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 <code>icon_flags</code> field of the <code>application_icon_descriptor</code> .

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](#).

11.7.3 Inter-Application communication API

The javadoc for the `org.dvb.io.ixc` package is provided in annex Y, "(normative): Inter-application communication API" on page 707. An example is provided in W.2, "Example of exporting an object for inter-application communication" on page 693.

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 extends, either directly or indirectly, 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 specified by the remote interface, and will contain no other members accessible to the application. These methods are called "remote methods." 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.

NOTE: The remote methods specified by a remote interface include all methods specified by that interface, including the methods inherited from superinterfaces. This applies even for methods inherited from superinterfaces that do not themselves extend `java.rmi.Remote`, either directly or indirectly.

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 methods specified by the declared remote interface type. A remote interface is an interface that extends `java.rmi.Remote`, either directly or indirectly. These remote methods must be declared 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 129.

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 208).

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 214).

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`.

11.8 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:

- | | |
|---|---|
| • <code>AccessControlContext</code> | • <code>Permissions</code> |
| • <code>AccessControlException</code> | • <code>Policy</code> |
| • <code>AccessController</code> | • <code>Principal</code> |
| • <code>AllPermission</code> | • <code>PrivilegedAction</code> |
| • <code>BasicPermission</code> | • <code>PrivilegedActionException</code> |
| • <code>CodeSource</code> | • <code>PrivilegedExceptionAction</code> |
| • <code>GeneralSecurityException</code> | • <code>ProtectionDomain</code> |
| • <code>Guard</code> | • <code>PublicKey</code> |
| • <code>InvalidKeyException</code> | • <code>SignatureException</code> |
| • <code>Key</code> | • <code>SecurityPermission</code> |
| • <code>KeyException</code> | • <code>SecureClassLoader (note 1)</code> |
| • <code>NoSuchAlgorithmException</code> | • <code>UnresolvedPermission</code> |
| • <code>NoSuchProviderException</code> | |
| • <code>Permission</code> | |
| • <code>PermissionCollection</code> | |

NOTE 1: The `java.security.SecureClassLoader` class extends the version of `java.lang.ClassLoader`, defined in [JAE 1.1.8 API \[31\]](#) therefore the `java.security.SecureClassLoader` class is not required to support additional APIs from later versions of `java.lang.ClassLoader`. It is an allowed implementation option for `SecureClassLoader` to be abstract.

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`

`X509Certificate` is not required to implement `X509Extension` on a compliant MHP terminal.

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 177.

11.8.3 Additional permissions classes

See T, "(normative): Permissions" on page 645.

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 331.

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 46 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 46 : 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"

Table 46 : System properties for profile and version interrogation (Sheet 2 of 2)

Property	Semantics	Possible values	Example
mhp.eb.version.minor	Minor version number of the supported enhanced broadcast profile	Non-negative integer value	"0"
mhp.eb.version.micro	Micro version number of the supported enhanced broadcast profile	Non-negative integer value	"0"
mhp.ib.version.major	Major version number of the supported interactive broadcast profile	Non-negative integer value	"1"
mhp.ib.version.minor	Minor version number of the supported interactive broadcast profile	Non-negative integer value	"0"
mhp.ib.version.micro	Micro version number of the supported interactive broadcast profile	Non-negative integer value	"0"
mhp.ia.version.major	Major version number of the supported internet access profile	Non-negative integer value	"1"
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 47 lists the currently defined options.

Table 47 : System properties for optional feature interrogation

Option	Property
IP Multicast over Broadcast Channel	mhp.option.ip.multicast
DSMCC user-to-user over the interaction channel	mhp.option.dsmcc.uu
DVB-HTML	mhp.option.dvb.html

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" .

`ServiceContextPermission("access", "own")` shall be granted to all MHP applications.

`ServiceContextPermission("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 157). 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.

NOTE: The `FilePermission` created shall comply with the access rights defined in section 12.6.2.7.2, "Policy for signed applications" on page 163.

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" if "read" is indicated in the credential; or string containing "write, delete" if "write" is 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

By default a `SelectPermission` is created with a locator "*" and action string "own" and a `ServiceContextPermission` is created with a name "*" and action string "own", unless denied by an entry in the permission request file.

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.

11.11 Content referencing

The following mapping shall be used between the types of locator defined in table 64, "Addressable entities, locators and their text representation" on page 213 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 64, "Addressable entities, locators and their text representation" on page 213 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`

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 136.
- `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 137.

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 dependent 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 208) 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

Where the HTTP (see 6.3.7.1, "HTTP 1.1" on page 49) protocol is supported, the following classes and methods shall support HTTP URLs as described below.

- 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)`

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.

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 157 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 11, "Value ranges for application_id" on page 81). 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 148.

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 14.

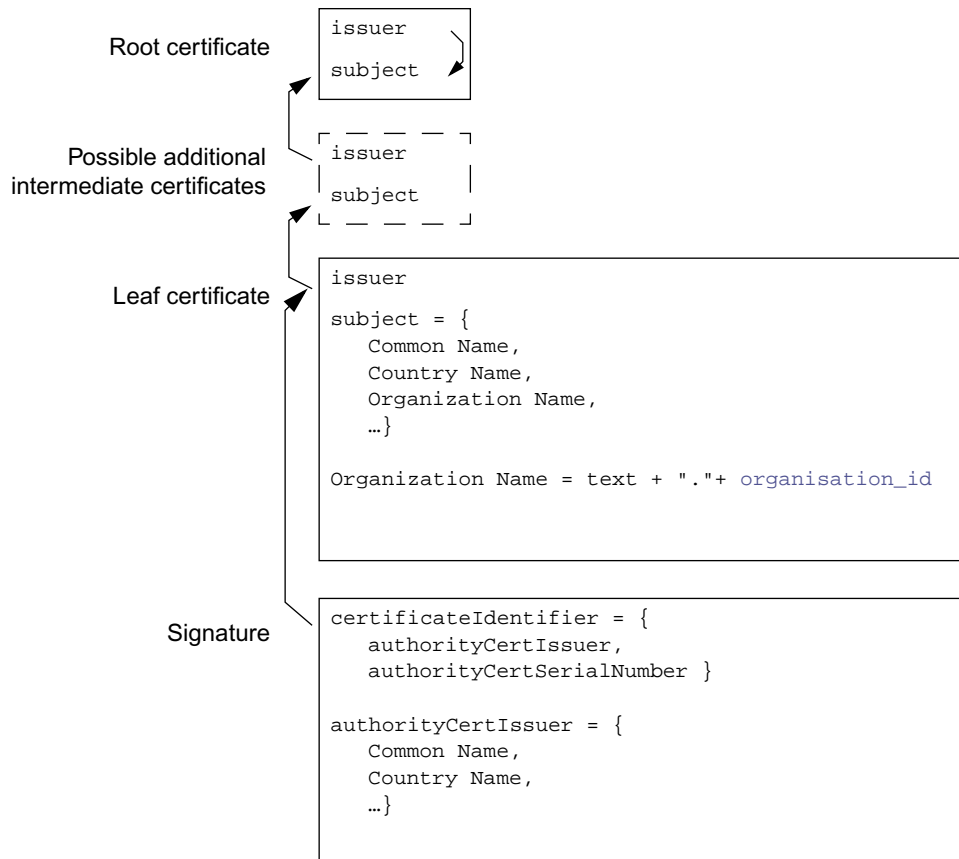


Figure 14 : Certificate chain illustrated

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, "MHP certification procedures" on page 171.

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 148
- "SignatureFile" on page 151
- "CertificateFile" on page 152

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. On transport protocols supporting the listing of files in a directory (e.g. Object Carousel), signature files shall not be listed. On transport protocols not supporting the listing of files in a directory (e.g. HTTP), signature files shall be listed and shall have a `digest_type` of non-authenticated. Those elements to be authenticated are associated with hashcodes. The syntax of the `HashFile` is shown in table 48.

Table 48 : Syntax of the Hashfile

Syntax	Num. Bits	Format
<pre> Hashfile () { digest_count for(i=0; i<digest_count; i++) { digest_type name_count for(j=0; j<name_count; j++) { name_length for(k=0; k<name_length; k++) { name_byte } } for(j=0; j<digest_length; j++) { digest_byte } } } </pre>	16	uimsbf
	8	uimsbf
	16	uimsbf
	8	uimsbf
	8	bslbf
	8	bslbf
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 computation rules and the digest algorithm, if any, used for the associated objects. Table 49 lists the allowed values for this field. The digest computation rules are defined in 12.4.1.3, "Digest value computation rules" on page 149.

Table 49 : Values of digest_type

value	digest len.	algorithm
0	0	Non authenticated
1	16	Digest computation rules without prefix and with MD-5 as defined in RFC 1321 [38]
2	20	Digest computation rules without prefix and with SHA-1 as defined in FIPS-180-1 [62]
3	20	Digest computation rules with prefix and with SHA-1 as defined in FIPS-180-1 [62].
Other values		Reserved for future use

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 49. MHP terminals shall support all digest algorithms.

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 149.

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:

"dvb.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 168.

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 (File or Directory) and on the value of [digest_type](#) and is specified in table 50.

Table 50 : Data required for digest value computation

Object type	Digest type	entry_type	Relevant data
File	1 or 2	not applicable	The entire content of the file
Directory			The content of the HashFile of the named directory
File	3	1	A prefix concatenated with the entire content of the file. The prefix is made of the entry_type encoded as a 32 bit uimsbf and concatenated with the file length in bytes encoded as a 32 bit uimsbf.
Directory		0	A prefix concatenated with the content of the HashFile of the named directory. The prefix is made of the entry_type encoded as a 32 bit uimsbf and concatenated with the file length in bytes encoded as a 32 bit uimsbf.
NOTE 1: All other values of entry type are reserved for future use			

12.4.1.3.1 Example

Consider two files `file1` and `file2` and one directory `dir1`. Using digest type 3, the digest value of `file1 + file2 + dir1` is equal to:

```
SHA-1 ( (uimsbf 32) 1 + (uimsbf 32) FileLength ( file1) + contents of file1
+ (uimsbf 32) 1 + (uimsbf 32) FileLength(file2) + contents of file2
+(uimsbf 32) 0 +(uimsbf 32) FileLength(HashFile(dir1))
+ contents of HashFile(dir1) ).
```

12.4.1.4 Warning concerning grouping of objects under a single digest (Informative)

Broadcasters should be made aware that during the construction of the object carousel, grouping objects under one digest can significantly and adversely affect the performance and memory requirements of an application on a terminal. For example, if 3 objects are grouped under the same digest, implementations must load all 3 objects even if only one is needed.

This specification in general recommends that while setting up a signed application for broadcast, only one digest be associated with each object requiring authentication. Grouping of multiple files under one hash value should only be employed if those files must always be loaded together and simultaneously. Additionally, the rate of change of the contents of an object should be carefully considered because grouping a rapidly changing object with more slowly changing objects for the purposes of hashing will negatively impact performance. When attempting such grouping, the resource limits of the MHP terminal should always be carefully considered.

12.4.1.5 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 neither named in the `HashFile` nor the `HashFile` itself, nor the signature files shall fail (that is behave as if the file is not available).

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 certificate 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.

NOTE: There is no security benefit in including a certificate file with a `digest_type` other than zero.

- h) When a permission request file exists, its entry in the `HashFile` cannot have a digest type of 0. MHP terminals shall ignore a permission request file with a digest type of 0.

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         OBJECT IDENTIFIER,
    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 180.

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 153.

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 153 and 12.4.4, "Integration" on page 153.

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 149.

12.4.2.5 Authentication rules

See [signatureValue](#) on page 151.

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 51.

Table 51 : Syntax of the CertificateFile

Syntax	Num. Bits	Format
<code>Certificatefile () { certificate_count for(i=0; i<certificate_count; i++) { certificate_length certificate() } }</code>	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 179.

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 154.

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 146, 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. However, for consistency, the file shall carry all of the certificate chain up to, and including, 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:

```
'dwb.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 151 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 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: The presence of more than one signature file enables more than one set of organisations to authenticate a subtree.

NOTE 3: When authenticating a signed application, an MHP terminal can select any one of these certificates to use to authenticate all subsequent classes or files loaded.

Where an MHP terminal trusts more than one of these certificates, it should attempt to select the most trusted of these. Apart from this, the mechanism used to make this selection is implementation dependent.

- 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) Use the corresponding certificate file to verify that the signature correctly signs the hash file.
- f) Follow the certificate chain contained within the certificate file verifying each link in the chain until the link to the root certificate is found.
- g) 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 103 for using the "same" leaf certificate to authenticate DVB-J class files shall be observed.

- h) 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 179 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 156.

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 152)
- The minimum set of attributes that shall be present in the subject are:
 - `commonName`
 - `countryName`
- If the certificate is a "leaf certificate" (see figure 14, "Certificate chain illustrated" on page 147) 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
```

Except for the presence of leading zeros, this is the same as 14.5, "Text encoding of application identifiers" on page 212.

This field reproduces the `organisation_id` value from the application's application identifier, see 10.5.1, "Encoding" on page 80. Before the application starts executing, the value of the `organisation_id` of at least one of the leaf certificates used in the authentication of the application's root subdirectory must match that in the application identifier. If this is not true, 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 326.

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 52 : 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 53 : Key for table 52

Keyword	When applied to ...	
	broadcasts	receivers
Mandatory	Certificates shall include these extensions.	Receivers shall observe the semantic for this information when provided.

Table 53 : Key for table 52

Keyword	When applied to ...	
	broadcasts	receivers
Optional	Certificates may or may 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 135 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 132).

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 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

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".

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)>

```

12.6.2.1.1 Number representation

12.6.2.1.1.1 Hex

Unless stated otherwise (e.g. section 14.5, "Text encoding of application identifiers" on page 212) 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 210) 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="0x4020">

  <file value="true"></file>

  <capermission>
    <casystemid
      id="0x1111" messagepassing="true"
      entitlementquery="true" mmi="false">
    </casystemid>
  </capermission>

  <applifecyclecontrol value="true"></applifecyclecontrol>

  <returnchannel>
    <defaultisp></defaultisp>
    <phonenumbers>+3583111111</phonenumbers>
    <phonenumbers>+3583111112</phonenumbers>
    <phonenumbers></phonenumbers>
  </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="0x05"></grantoridentifier>
    <expirationdate date="24/12/2032"></expirationdate>
    <filename read="true" write="false">5/15/dirl/scores</filename>
    <filename read="true" write="false">5/15/dirl/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 54 for the types defined in this specification.

Table 54 : 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 34 on page 96
0x0002	DVB-HTML	The name <i>initial_path_bytes</i> , see table 36 on page 97

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 12.6.2.1.1.1, "Hex" on page 159.

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 12.6.2.1.1.1, "Hex" on page 159.

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 file 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 directory containing the permission request file shall be searched first and then recursively ascend the directory hierarchy checking certificate files until one is found with the file name matching the contents of the `certchainfileid` field of the credential. If the root of the file system is reached without finding a matching certificate then the file access shall not be granted

The `grantoridentifier` in the `persistentfilecredential` shall match the `organisation_id` contained in the `Subject organisationName` field of the leaf certificate. Likewise, the `grantoridentifier` shall match the organisation id of the owner of the file to which access is granted by this `persistentfilecredential`.

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 55 :

Elements	Comments
grantoridentifier	<p>This element contains the 32 bit organization id identifying the grantor organization. The CDATA attribute id of this element shall have the following BNF syntax: <code>"0x" 0*8hex ; e.g. 0x34243</code></p> <p>See 12.6.2.1.1.1, "Hex" on page 159.</p>
expirationdate	<p>This element contains the expiration date of this credential. The implementation should ignore the certificate if the date has expired. The CDATA attribute date shall follow the following BNF syntax: <code>2dec "/" 2dec "/" 4dec ; e.g. "01/12/2001"</code></p> <p>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.</p>
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: <code>1*(alphanum "*" "." "-" "_" "?")</code> <code>1*(alphanum "/" "*" "." "-" "_" "?")</code></p> <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 read and write 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 151.</p> <p>The signature is encoded using the Base64 content transfer encoding as specified in section 6.8 of RFC 2045 [64].</p>
certchainfileid	<p>This element contains the identifier of the leaf certificate, i.e. the "x" in the file name "dvb.certificates.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 56 :

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<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 152. 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 Policy for signed applications

No access to the persistent storage, unless otherwise indicated in the Permission File.

When the permission request file requests access to persistent storage and this is granted, 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 122. 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 212. By default, files created by an application will have owner read/write access only.

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 K, "(normative): DVB-J persistent storage API" on page 360.

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.addMMIListener
- 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.addMMIListener
- 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 12.6.2.1.1.1, "Hex" on page 159.

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 165.

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"
```

The presence of the `defaultisp` tag indicates that the application is allowed to use the default ISP connection. If this tag is not present, then the application shall not be able to specify connecting to the default target except where the connection parameters for that target are explicitly specified by the application and are included in the phone number string.

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 (see 12.6.1, "General principles" on page 157).

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.

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) 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 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 specify the host and the action shall be formatted as specified in the `java.net.SocketPermission` class as defined in 11.8, "Security" on page 132.

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"
>
```

12.7 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 an application 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/fool.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:

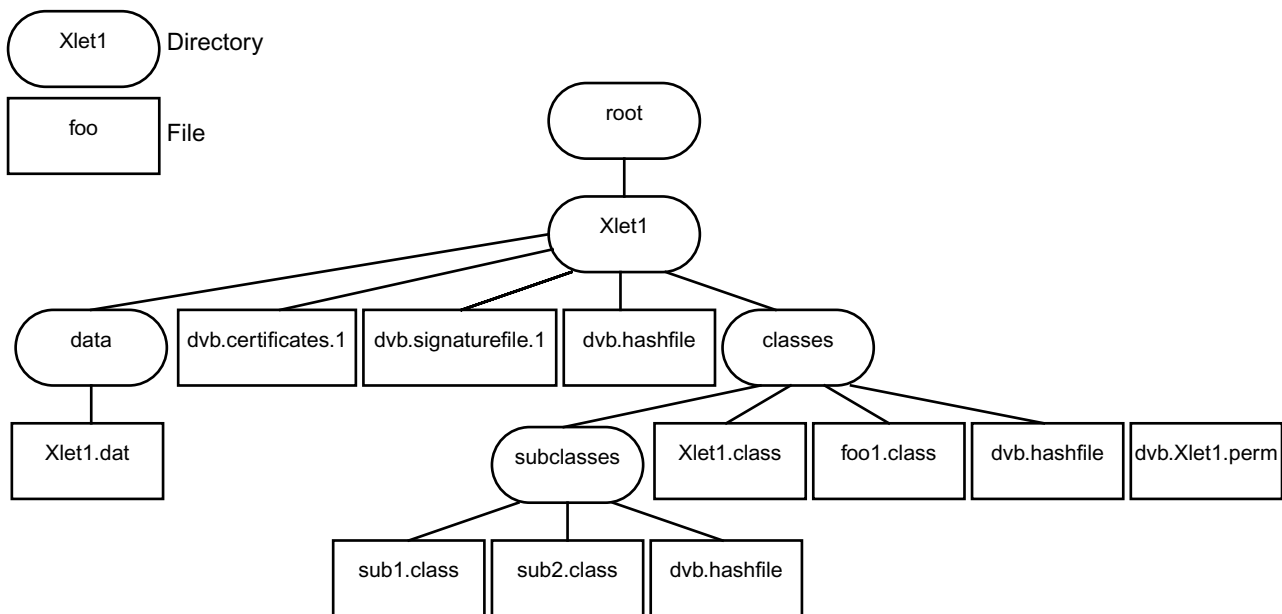


Figure 15 : Example of an authenticated file tree

12.7.2 Hashes and signature computations:

12.7.2.1 Computation of the hashes of the root/Xlet1/classes/subclasses directory

- initialise the MD5 algorithm
- 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 H_0
- construct the contents of the hash file for the directory `root/Xlet1/classes/subclasses/` as follows:

Table 57 : 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	
H_0	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.

- 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

- initialise the MD5 algorithm, compute the MD5 hash H_2 using the contents of the file `root/Xlet1/classes/subclasses/dvb.hashfile`
- initialise the MD5 algorithm, compute the MD5 hash H_3 using the contents of the file `root/Xlet1/classes/Xlet1.class`
- initialise the MD5 algorithm, compute the MD5 hash H_4 using the contents of the file `root/Xlet1/classes/foo1.class`

h) construct the contents of the hash file for the directory `root/Xlet1/classes/` as follows:

Table 58 : 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 subclasses 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 Xlet1.class
1	Type of digest algorithm = MD5
1	Number of entries over which a MD5 hash has been computed
foo1.class	List of names of entries
H4	MD5 hash of file foo1.class

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 59 : 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 classes 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.signaturefile.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 MHP certification procedures

The MHP certification procedures are outside of the scope of this document.

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.

The number of CRLs to be supported per level is defined in G, "(normative): Minimum Platform Capabilities" on page 326.

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 or the information they contain (including serial numbers) 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, "Platform minima" on page 187](#).

12.9.1.3.3 Storage management

The broadcast CRL and RCMM signalling (see [12.9.2, "Root certificate management" on page 175](#)) 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.

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 153](#).

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 179](#):

```

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 154. This applies to the following fields:

signatureAlgorithm : follows "signatureAlgorithm" on page 154.

signatureValue: follows "signatureValue" on page 154.

version: follows "version" on page 154.

signature: follows "signatureAlgorithm" on page 154.

issuer: follows "issuer" on page 154.

thisUpdate: Publication date of this CRL. Follows the encoding of Time used for **validity**. See "validity" on page 155.

nextUpdate: Publication date of the next version of the CRL. Follows the encoding of Time used for **validity**. See "validity" on page 155.

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 155.

crlExtensions: The syntax of the **Extensions** element is reproduced on "Extensions" on page 183. 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 184.

The following crlExtension shall be supported:

- AuthorityKeyIdentifier as defined in section 12.4.2.1 on page 151.

This extension identifies the certificate that is needed to check the signature of the CRL.

The crlExtension **AuthorityKeyIdentifier** shall be included in every CRL. MHP implementations shall use the **AuthorityKeyIdentifier** to find the certificate that carries the public key that is used to check the signature of the CRL and ignore any CRLs where this check fails or where the **AuthorityKeyIdentifier** is not present.

Other crlExtensions are optional.

12.9.1.9 CRL Processing

CRLs, or the information they contain (including serial numbers), shall be kept in persistent storage in the MHP terminal.

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 153, 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) The signature of the CRL is verified and the signing certificate is verified to be from a trusted source. If the certificate cannot be trusted, the CRL is ignored.
- 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, "Platform minima" on page 187.

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:

```
"dvb.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, "Platform minima" on page 187](#). 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.

12.9.3 Test certificates

There shall be a means to make test root certificates (as required by [12.12, "Platform minima" on page 187](#)) available when the terminal is in test mode. The method for doing so shall be implementation dependent; it may involve either replacing non-test root certificates, or storing test root certificates in addition to non-test root certificates.

Test root certificates shall only be functional while the terminal is in test mode. Objects signed with test root certificates shall not be used other than for the purposes of testing.

This specification is intentionally silent on the test root certificates to be used.

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 60 (see RFC 2246 [63] for definition of the terms) which identifies which methods are required in an MHP.

Table 60 : Profile of cipher suites that implementations are required to support

CipherSuite	Key Exchange	Cipher	Hash	Value (hex)	MHP status
TLS_NULL_WITH_NULL_NULL (note 1)	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
NOTE 1: This cipher suite is only used by a TLS implementation during the negotiation of a connection. It is not required to be enabled as a cipher suite that is available for a negotiated connection.					

NOTE: The ciphers TLS_RSA_WITH_NULL_MD5 and TLS_RSA_WITH_NULL_SHA provide integrity checking but without confidentiality (i.e. data are in clear). Data encryption is very time consuming for a server. They are useful for applications in which data don't need to be encrypted but in which data integrity is very important. For these applications, the server will only have to compute a HMAC for every message exchanged.

12.10.3 Downloading of certificates for TLS

12.10.3.1 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 133. The process of server authentication involves the checking of the certificate chain sent by the TLS server.

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).

12.10.3.2 Usage of certificate in TLS

12.10.3.2.1 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.

12.10.3.2.1.1 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 212.

Location of the TLS certificates is application type dependant. See table 61.

Table 61 : 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 96.
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 97.
0x0003...0xFFFF	reserved for future use

12.10.3.2.1.2 Certificate authentication

To be considered valid TLS certificates the certificate files shall be authenticated members of the same authenticated subtree as the application.

12.10.3.2.2 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 133) 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 154.

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 154.

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 154.

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 154.

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 154.

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 }

```

A non-root certificate validity is limited by the validity period of the signer's certificate.

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 155.

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 181](#)). 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 an EmailAddress 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 179) 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 155.](#)

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 155.](#)

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 156.](#)

12.11.2 Standard certificate extensions

See table 52, "Profile for standard certificate extensions" on page 156.

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 Platform minima

NOTE: At the time of writing, the minima described in this section have not been tested in real operations. It is believed possible that these minima may need to be revised upwards in future versions of this specification.

MHP platform hardware is required to support the following minimum sizes to support the MHP security model:

- A value of 2 for N in [12.9.2.2, "Security of RCMM" on page 175](#)
- A minimum depth of 5 levels in the certificate chain
- A minimum of 5 CRLs
- A minimum of 10 entries per CRL
- A minimum of 3 root certificates (regardless of the number of underlying root certificate authorities).
- A minimum of 3 root certificates for testing (see [12.9.3, "Test certificates" on page 177](#)).

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.

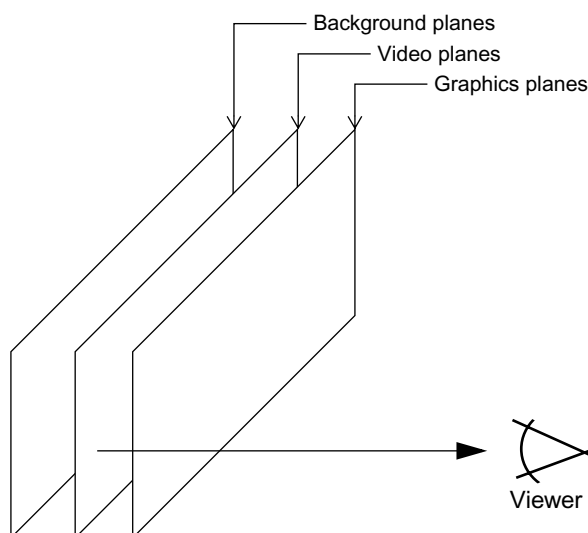


Figure 16 : Illustration of the different types of display planes

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 204.

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 196). 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 326.

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).

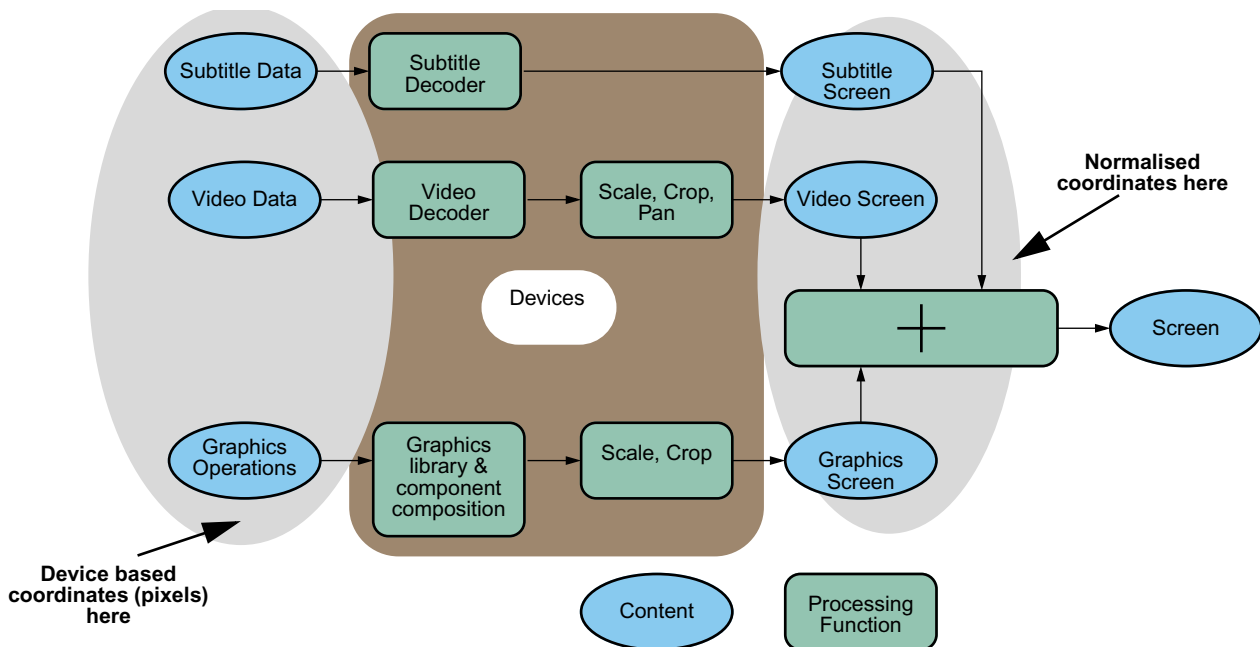


Figure 17 :

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

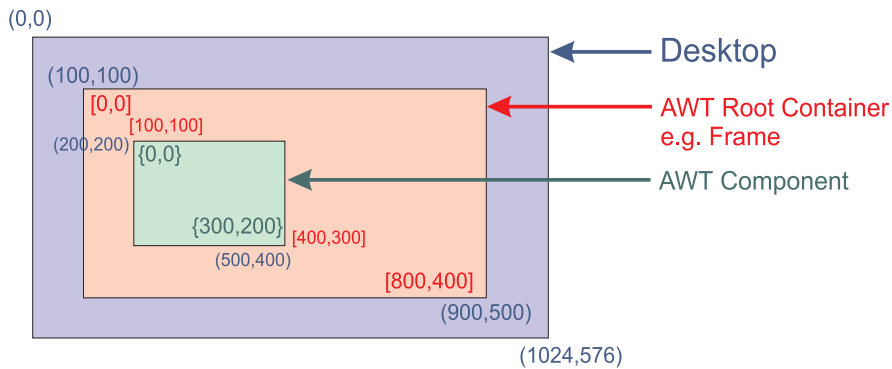
A normalised screen coordinate system supports references to positions and sizes in the video output from an MHP device without reference to any form of pixels.

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 18 : AWT Coordinate system in a computer environment

Figure 18 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 18).

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.

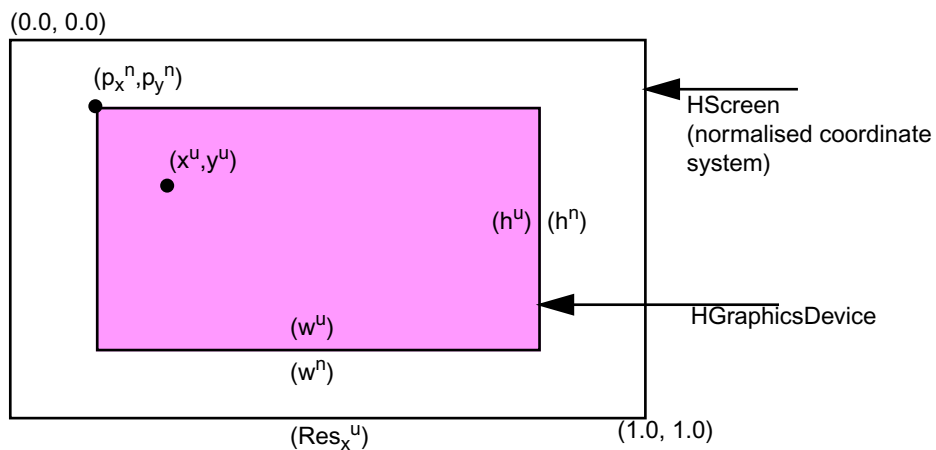


Figure 19 : Conversion between normalised and user coordinate systems

(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 $w^u \times h^u$ than 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 (xⁿ,yⁿ) 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.

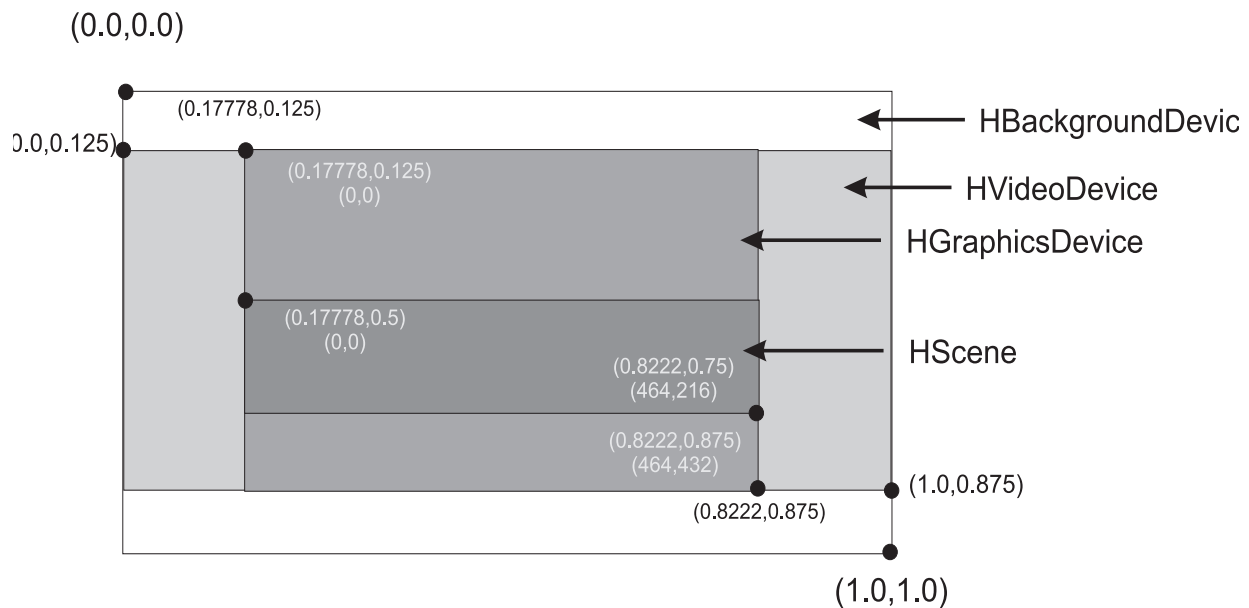


Figure 20 : Possible configuration of HAVi Devices

Figure 20 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 20 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 21

$$AR^{pixel} = \frac{AR_x^{pixel}}{AR_y^{pixel}} = \frac{AR_x^{display} \cdot w^n}{AR_y^{display} \cdot h^n} \cdot \frac{h^u}{w^u}$$

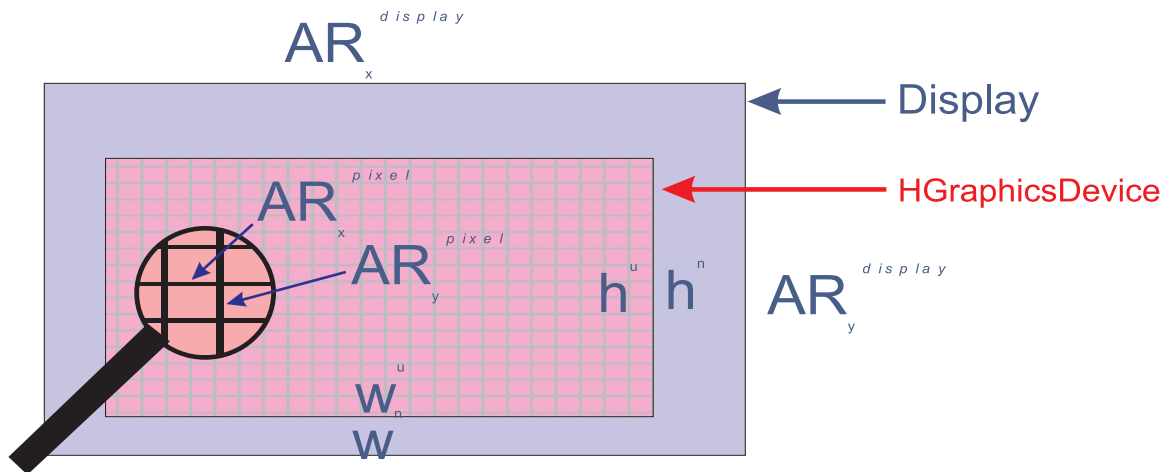


Figure 21 : Calculating the Pixel Aspect Ratio

Table 62 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 326.

Table 62 : Typical Resolutions and their pixel aspect ratio (informative)

	4:3 Display	16:9 Display
Resolution for full screen HGraphicsDevice	Pixel Aspect Ratio	Pixel Aspect Ratio
720x576 (note 1)	1150:1053	4600:3159
768x576	1:1	48:36
1024x576	36:48	1:1
NOTE 1: Based on 702 nominally displayable pixels derived from ITU-R BT.470 [28] and ITU-R BT.601 [29]. This is approximate due to the tolerances defined.		

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 205. The "Graphics, Video and Subtitle pipeline" is illustrated in figure 22.

Figure 22 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.

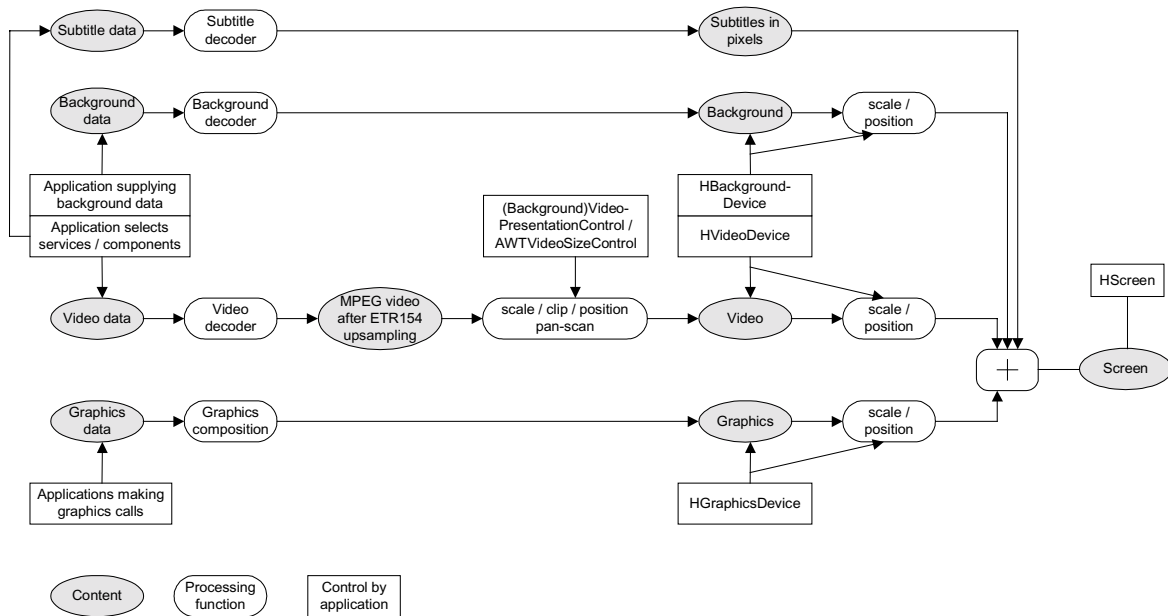


Figure 22 : Graphics, Video and Subtitle pipeline

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_OVER 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 23.

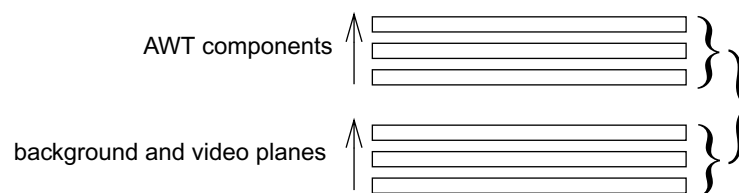


Figure 23 : Overview of AWT / HAVi plane composition

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 24) 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 24.

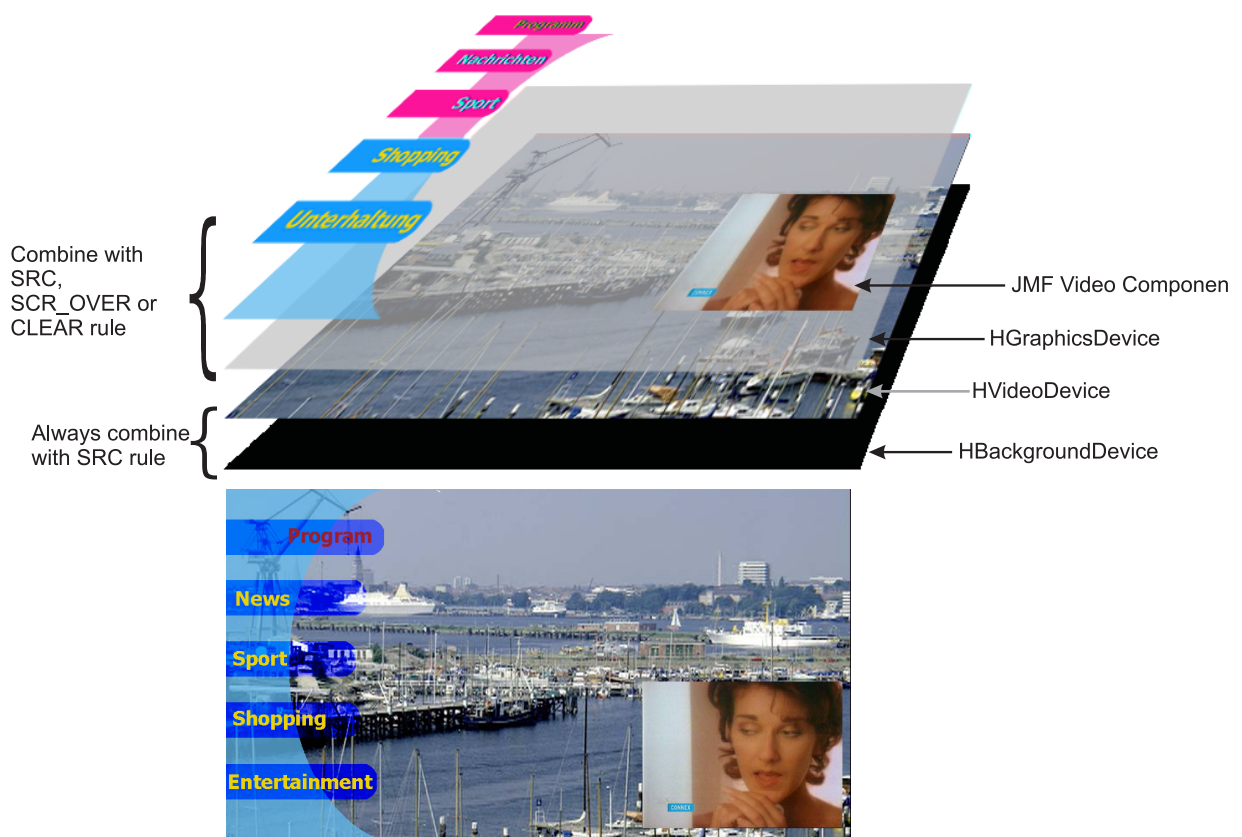


Figure 24 : The MHP display stack illustrated

MHP terminals are allowed to support only non overlapping HScenes. In implementations not supporting overlapping HScenes the 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 26. 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.

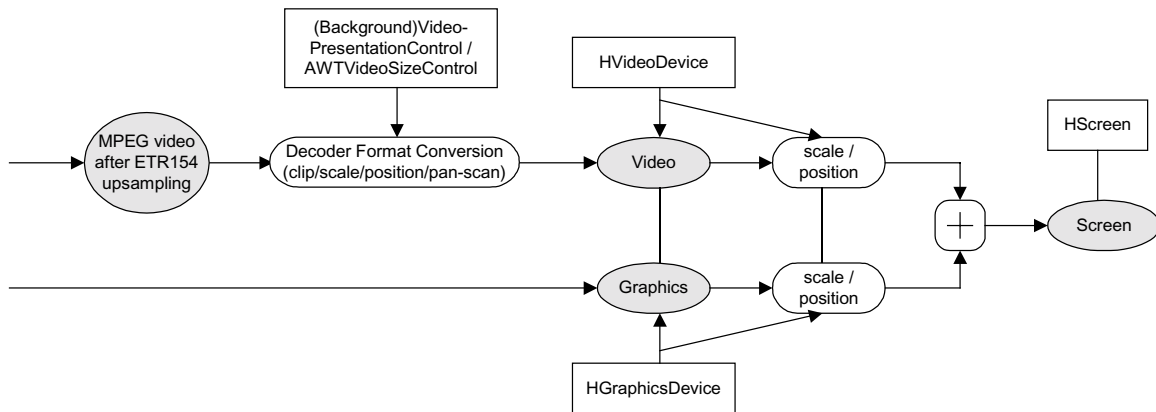


Figure 26 : Video and graphics pixel impact

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 27. 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]).

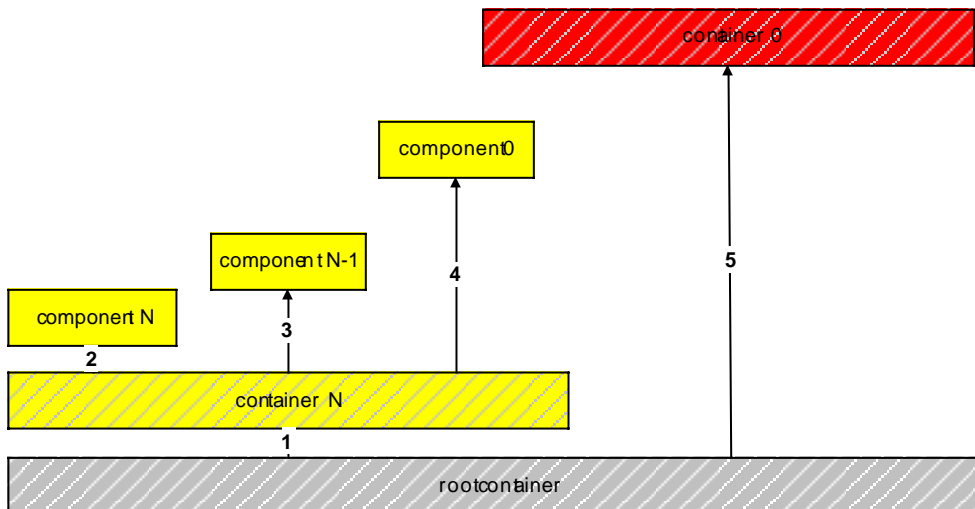


Figure 27 : Chronological order of painting

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 28 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 28 : 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 [JAE 1.1.8 API \[31\]](#). However, certain extensions are made.

In order to allow compositing in the MHP the graphics capabilities of [JAE 1.1.8 API \[31\]](#) 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 653](#).

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. All images whose `getGraphics` method returns a non-null object shall support alpha.

The supported compositing rules are defined in [G, "\(normative\): Minimum Platform Capabilities" on page 326](#).

The default compositing rule used by all graphics objects is `SRC_OVER`.

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 36 or by approximating the alpha, see [G, "\(normative\): Minimum Platform Capabilities" on page 326](#)) 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 29. 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.

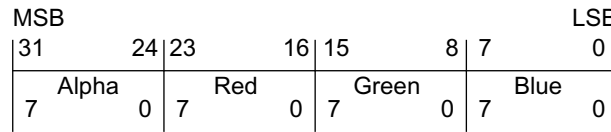


Figure 29 : MHP colour format (TYPE_INT_ARGB)

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 52).

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 30 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)

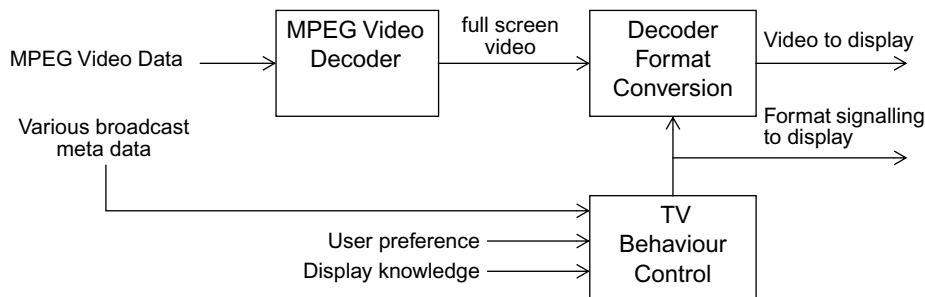


Figure 30 : Format control in TV mode

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 31. In addition, there are two alternative sources of control for the "Decoder Format Conversion" process:

- Conventional TV format control behaviour (as in figure 30)
- 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.

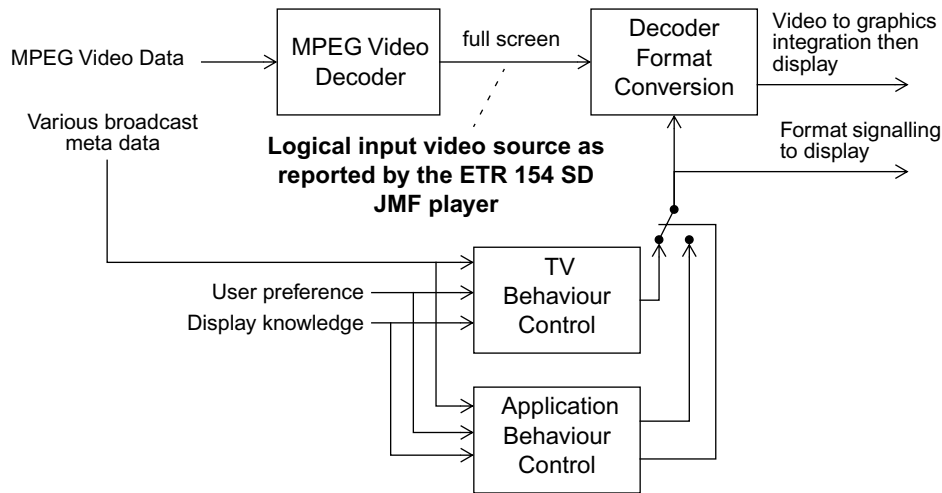


Figure 31 : Format control in the presence of a JMF player

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 32, these steps are clipping, scaling and positioning.

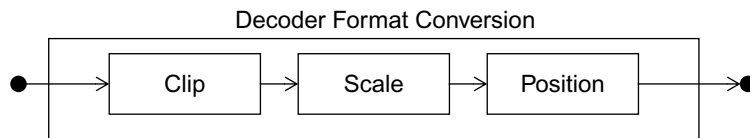


Figure 32 : Reference model for Decoder Format Conversion

An application can query the implemented capabilities of each step of the Decoder Format Conversion using the `org.dvb.media.VideoPresentationControl` ("VideoPresentationControl" on page 502). 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 507) 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 TR 101 154 [9] 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 465) which allows an arbitrary portion of the video to be placed within the AWT hierarchy.

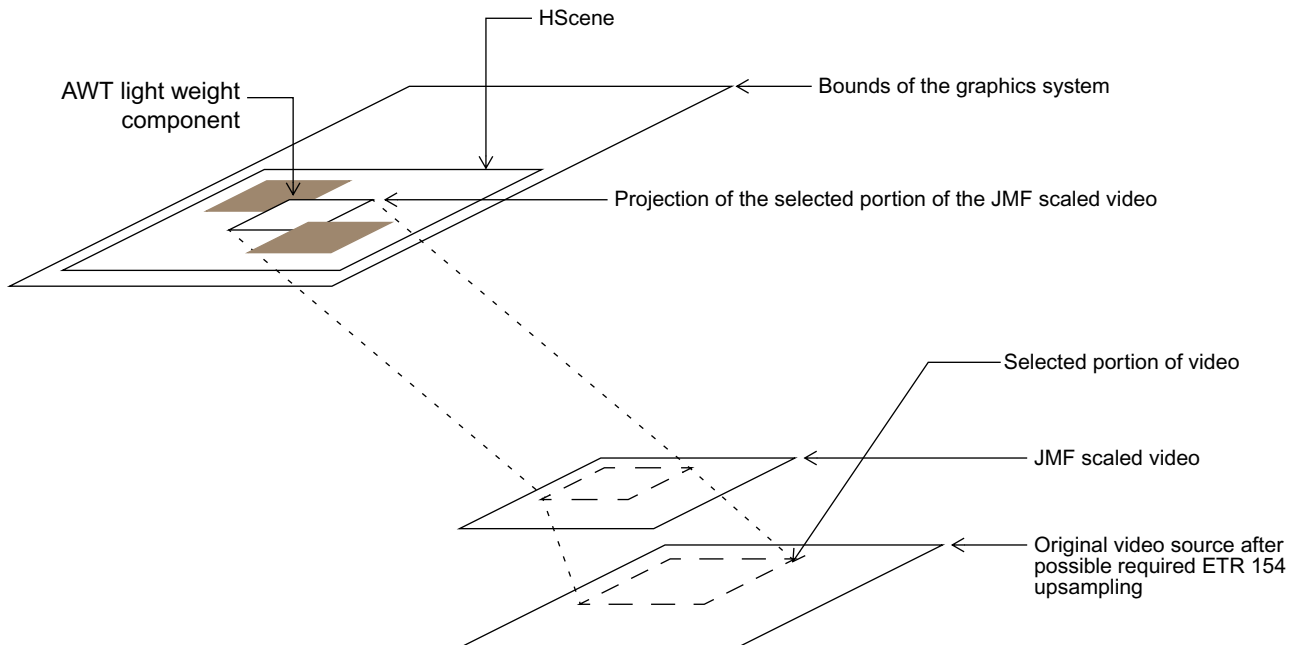


Figure 33 : Introducing video into the AWT component stack

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:

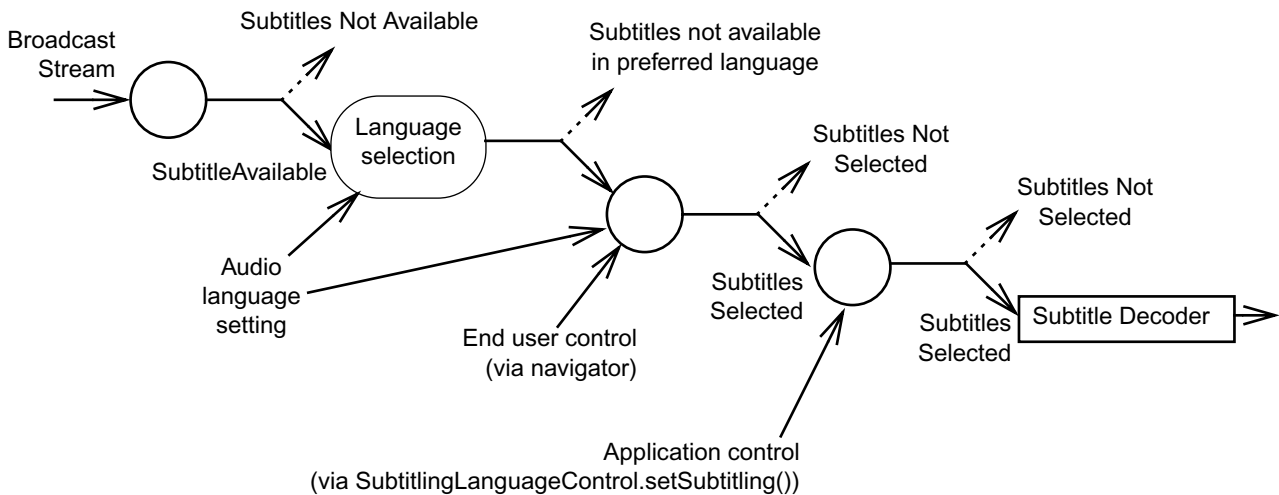


Figure 34 : Determining subtitling language and presentation setting

The DVB-J API includes a control (see "[SubtitlingEventControl](#)" on page 494) 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 language selection and the resulting preferred language are determined using an implementation dependent algorithm. e.g. the platform may support separate language preferences for audio and for subtitles.

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 minimum set of operations required is defined in ([G.1.3.1, "Composition rules" on page 327](#)), in addition the implementation of the compositions may be approximated as detailed below.

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 35](#)). This places certain practical limitations on the purity of the display model.

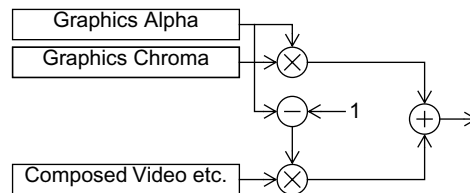


Figure 35 : Typical current technology implementation

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 video:

- 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 when $0 < \alpha < 1$. Drawing with colour where $\alpha = 0$ (i.e. fully transparent) shall not cause any effect to the existing pixels. [Figure 36 on page 206](#) illustrates a variety of implementations of SRC_OVER graphics to graphics blending. One allowed behaviour where the sample model is of TYPE_BASE is that defined in [JAE 1.1.8 API \[31\]](#) for those signatures of the drawImage methods in java.awt.Graphics which do not have a "bgColor" parameter. Specifically, only pixels which are fully opaque are drawn at all. Pixels which are transparent to any extent do not affect whatever pixels are already there.

- [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.

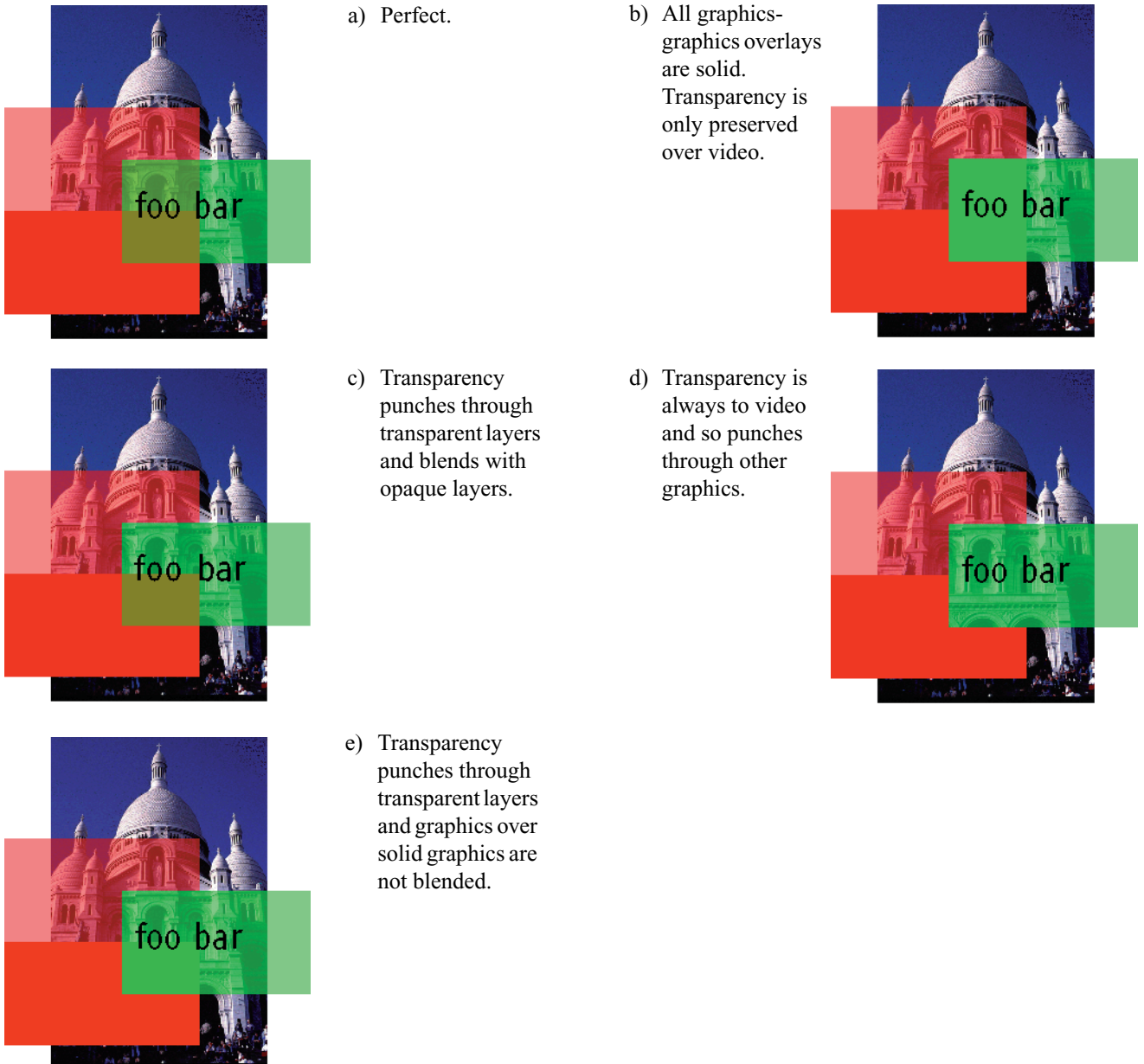


Figure 36 : Implementations of blending

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.

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 326. 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 327.

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 63 : 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 RFC 2396 [41])	

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 52).
- 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 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 incorrectly formatted data, however transmitted or otherwise acquired, is implementation dependent except where a specific error behaviour is required by this specification or referenced specifications. 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.

MHP terminals should observe the behaviour defined in section 4.1 of TR 101 154 [9].

NOTE: It is highly recommended that the MHP terminal should be designed to allow for future compatible extensions to the DVB SI, DSMCC or other formatted data interpreted by the MHP terminal. All of the fields "reserved" (for ISO), "reserved_future_use" (for ETSI), and "user defined" in the EN 300 468 [4] should be ignored by MHP terminals not designed to make use of them. The "reserved" and "reserved_future_use" fields may be specified in the future by the respective bodies, whereas the "user defined" fields will not be standardized. Where an MHP API provides access to this data, the data should be returned to the MHP application without validation or correction.

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 118 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`.

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 64 : 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

Table 64 : Addressable entities, locators and their text representation (Sheet 2 of 2)

Entity	Text Representation
File	"file:" URL as defined in RFC 1738 [67] (note 1) DVB locator including "dwb_abs_path" element (note 2)
Directory	"file:" URL as defined in RFC 1738 [67] (note 1) DVB locator including "dwb_abs_path" element (note 2)
Drip feed decoder	"dripfeed://"
<p>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.</p> <p>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.</p>	

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 113.

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 101).

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.

14.10 CA system

In this clause the term "CA system" applies to the CA system however implemented and thus embraces both embedded CA systems and those implemented via the DVB common Interface.

14.10.1 Service selection

Where the CA system causes changes in the currently decoded service the effect is equivalent to a service selection.

In the specific case of the DVB Common Interface this will occur when a Host Control tune request is made by a module.

14.10.2 Media component selection

Requests from the CA system to change in composition of the media components (Audio, Video or subtitles) shall be automatically performed (this effect is equivalent effect to using `javax.tv.media.MediaSelectControl`) and shall be reported to the application by the `org.davic.net.ca.PIDChangeEvent`.

In the specific case of the DVB Common Interface this will occur when a Host Control replace / clear_replace request is made by a module.

14.10.3 Non-media component selection

Requests from the CA system to change in composition of the non-media components (e.g. DSM-CC Object Carousel and private data) shall not be automatically performed but shall be reported to the application by the `org.davic.net.ca.PIDChangeEvent`.

In the specific case of the DVB Common Interface this will occur when a Host Control replace / clear_replace request is made by a module.

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 65 : Detailed platform profile definitions (Sheet 1 of 2)

Area	Specification	Enhanced Broadcast Profile 1	Interactive Broadcast Profile 1	Internet Access Profile 1
Static formats				
Bitmap pictures	7.1.1.3, "PNG" on page 50 + 15.1, "PNG - restrictions" on page 218	M	M	
	7.1.1.3, "PNG" on page 50 without restrictions	-	-	
	7.1.1.4, "GIF" on page 50	-	-	
	7.1.2, "MPEG-2 I-Frames" on page 50	M	M	
	7.1.1.2, "JPEG" on page 50 + 15.3, "JPEG - restrictions" on page 218	M	-	-
	7.1.1.2, "JPEG" on page 50 without restrictions	-	M	
Audio clips	7.1.4, "Monomedia format for audio clips" on page 52	M	M	
Video drips	7.1.3, "MPEG-2 Video "drips"" on page 50	M	M	
Text encoding	7.1.5, "Monomedia format for text" on page 52	M	M	
Broadcast streaming formats				
Video	7.2.2, "Video" on page 52	M	M	
Audio	7.2.1, "Audio" on page 52	M	M	
Subtitles	7.2.3, "Subtitles" on page 52	M	M	
Fonts				
Built in	Character set see annex E, "(normative): Character set" on page 320, Metrics see annex D, "(normative): Text presentation" on page 302 Face: UK RNIB 'Tiresias'	M	M	
Downloadable	7.4, "Downloadable Fonts" on page 53	M	M	
Broadcast channel protocols				
	6.2.2, "MPEG-2 Sections" on page 46	M	M	
	6.2.5, "DSM-CC User-to-User Object Carousel" on page 46	M	M	
	IP Multicast stack based on: 6.2.6, "DVB Multiprotocol Encapsulation" on page 47, 6.2.7, "Internet Protocol (IP)" on page 47 6.2.8, "User Datagram Protocol (UDP)" on page 47	O	Ro	M
Interaction channel protocols				
TCP/IP	6.3.3, "Transmission Control Protocol (TCP)" on page 48 6.3.2, "Internet Protocol (IP)" on page 48	-	M	M
UDP/IP	6.2.8, "User Datagram Protocol (UDP)" on page 47 6.3.2, "Internet Protocol (IP)" on page 48	-	M	M
DSM-CC U-U RPC	6.2.5, "DSM-CC User-to-User Object Carousel" on page 46 6.3.4, "UNO-RPC" on page 48 6.3.5, "UNO-CDR" on page 48	-	O	
HTTP	6.3.7.1, "HTTP 1.1" on page 49	-	O	M
DVB-J				
Core	11.3, "Fundamental DVB-J APIs" on page 104	M	M	
Presentation	11.4.1, "Graphical User Interface API" on page 109	M (note 1)	M (note 1)	
	11.4.2, "Streamed Media API" on page 113	M	M	

Table 65 : Detailed platform profile definitions (Sheet 2 of 2)

Area	Specification	Enhanced Broadcast Profile 1	Interactive Broadcast Profile 1	Internet Access Profile 1
Data Access	11.5.1, "Broadcast Transport Protocol Access API" on page 118	M	M	
	11.5.2, "Support for Multicast IP over the Broadcast Channel" on page 120	O	Ro	
	11.5.3, "Support for IP over the Return Channel" on page 121	-	M	
	11.5.4, "MPEG-2 Section Filter API" on page 121	M	M	
	11.5.5, "Mid-Level Communications API" on page 121	-	M	
	11.5.6, "Persistent Storage API" on page 122	M	M	
Service Information & Selection	11.6.1, "DVB Service Information API" on page 123	M	M	
	11.6.2, "Service Selection API" on page 123	M	M	
	11.6.3, "Tuning API" on page 124	M	M	
	11.6.4, "Conditional Access API" on page 125	M	M	
	11.6.5, "Protocol Independent SI API" on page 125	M	M	
Common Infrastructure	11.7.1, "APIs to support DVB-J application lifecycle" on page 126	M	M	
	11.7.2, "Application discovery and launching APIs" on page 127	M	M	
	11.7.3, "Inter-Application communication API" on page 128	M	M	
	11.7.4, "Basic MPEG Concepts" on page 130	M	M	
	11.7.5, "Resource Notification" on page 131	M	M	
	11.7.6, "Content Referencing" on page 131	M	M	
	11.7.7, "Common Error Reporting" on page 132	M	M	
Security	11.8.1, "Basic Security" on page 132	M	M	
	11.8.2, "APIs to Support TLS / SSL Over the Return Channel" on page 133	-	M	
	11.8.3, "Additional permissions classes" on page 133	M	M	
Others	11.9.1, "Timer Support" on page 134	M	M	
	11.9.2, "User Settings and Preferences API" on page 134	M	M	
	11.9.3, "Profile and version properties" on page 134	M	M	
NOTE 1: The javax.tv.graphics.TVContainer.getRootContainer method shall return an instance of org.havi.ui.HScene or null.				

Key	
-	Not required / Not applicable
O	Optional feature in the receiver
Ro	Recommended optional feature in the receiver
M	Mandatory feature in the receiver

15.1 PNG - restrictions

MHP terminals are required to support ALL of the PNG colour types defines in PNG Specification Version 1.0 (see table 66). MHP terminals are responsible for mapping these colours to those used by the terminal's OSD.

Any combination of PNGs with different colour types may be active at any one time. Similarly, terminals are responsible for mapping RGB16 direct colour specifications to colours that the OSD can support.

Table 66 : 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.

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 67 : Media type support required in Enhanced and Interactive Broadcast profile 1

Media type	Reference	API(s)
Downloadable fonts	7.4	org.dvb.ui.FontFactory
Audio from file	7.1.4	org.havi.ui.HSound JMF only with references to files
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	EN 300 468 [4]	JMF only with references to DVB services for playback direct from the network
Video "drips"	7.1.3	org.dvb.media.DripFeedDataSource
Text files	7.1.5	All Java APIs supporting reading or writing text files.
Subtitles	7.2.3	JMF only as part of a DVB service

15.3 JPEG - restrictions

The restricted JPEG specification is as specified in 7.1.1.2, "JPEG" on page 50 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.

15.5 Video raster format dependencies

This section addresses the aspects of this specification that vary as a consequence of the video raster format. The formats names follow those used in [TR 101 154 \[9\]](#).

15.5.1 25 Hz standard definition

15.5.1.1 Logical pixel resolution

The logical pixel resolution shall be 72 dots per inch.

16 Registry of Constants

16.1 System constants

Table 68 : 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 134.
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 134.

Table 69 : Profile encoding

application profile	version			Definition
	major	minor	micro	
1	1	0	2	Enhanced Broadcast Profile 1 as defined in this specification
2	1	0	2	Interactive Broadcast Profile 1 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.DVBJProxy.LOADED = 5;

public static final int org.dvb.application.DVBHTMLProxy.LOADING=6;
public static final int org.dvb.application.DVBHTMLProxy.KILLED=7;

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 int org.dvb.net.rc.RCInterface.TYPE_RCS = 7;

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.

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.1.6 `java.io.PrintStream`

The two constructors `PrintStream(OutputStream)` and `PrintStream(OutputStream, boolean)`, are considered to not be deprecated, and are therefore required to be present on an MHP terminal.

A.1.7 `java.io.Serializable`

NOTE: This specification does not have standardised external forms for serializable classes. Thus applications cannot rely on using serialization for transmitting data into or out of a specific MHP terminal. Use of `Serialized` objects with persistent storage shall work as specified where applications have the required permissions however using this is not recommended.

A.1.8 `java.io.ObjectStreamConstants`

The interface `java.io.ObjectStreamConstants` implemented by `java.io.ObjectInputStream` is not part of this specification.

A.1.9 java.net.SocketOptions

The interface `java.net.SocketOptions` implemented by `java.net.SocketImpl` is not part of this specification.

A.1.10 java.util.zip.ZipConstants

The interface `java.util.zip.ZipConstants` implemented by `java.util.zip.ZipEntry`, `java.util.zip.ZipFile` and `java.util.zip.ZipInputStream` is not part of this specification.

A.1.11 Component

The specification of `Component.getGraphics` in [JAE 1.1.8 API \[31\]](#) is considered to read:

```
/**
 * Creates and returns a graphics context to be used for
 * rendering directly to this component.
 * Returns <code>null</code> if this component is currently not
 * displayable.
 * The returned object is a graphics context that causes drawing to the output device of this
component;
 * in other words, it is the same whether or not
 * double buffering is enabled on this component.
 * When you have finished using the graphics context
 * returned by this method,
 * you should invoke the <code>dispose</code> method on it.
 *
 * @return a graphics context for this component, or <code>null</code>
 *         if it has none
 * @see     #paint
 * @see     Graphics#dispose
 * @since   JDK1.0
 */
```

A.1.12 java.awt.event.KeyEvent

The constructor:

```
KeyEvent(Component, int, long, int, int)
```

shall be considered to be deprecated.

A.1.13 java.awt.Component

A.1.13.1 java.awt.Component.update(Graphics)

The following text:

The AWT calls the update method in response to a call to `repaintupdate` or `paint`. You can assume that the background is not cleared.

Is considered to be replaced by:

If this component is not a lightweight component, the AWT calls the update method in response to a call to `repaint`. You can assume that the background is not cleared.

A.1.13.2 java.awt.Component.repaint()

The following text:

This method causes a call to this component's update method as soon as possible.

Is considered to be replaced by:

If this component is a lightweight component, this method causes a call to this component's `paint` method as soon as possible. Otherwise, this method causes a call to this component's update method as soon as possible.

A.1.13.3 `java.awt.Component.repaint(long)`

The following text:

Repaints the component. This will result in a call to update within *tm* milliseconds.

Is considered to be replaced by:

Repaints the component. If this component is a lightweight component, this will result in a call to paint within *tm* milliseconds.

Otherwise, this will result in a call to update within *tm* milliseconds.

A.1.13.4 `java.awt.Component.repaint(int, int, int, int)`

The following text:

This method causes a call to this component's update method as soon as possible.

Is considered to be replaced by:

If this component is a lightweight component, this method causes a call to this component's paint method as soon as possible. Otherwise, this method causes a call to this component's update method as soon as possible.

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.3.2 `javax.media.protocol.ContentDescriptor`

A.3.2.1 `getContentTypes`

This method shall return the string passed into the constructor; which is defined to be a MIME type.

A.3.2.2 `contentTypeToPackageName`

This method is considered to have "public" visibility.

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.5.5 `javax.tv.graphics.AlphaColor`

The following features imply dependencies on Java 2 and are therefore not considered to be present in this specification:

- the `createContext` method listed under "Methods inherited from `java.awt.Color`"
- the `java.awt.Transparency` interface listed under "All Implemented Interfaces:"
- the fields listed under "Fields inherited from interface `java.awt.Transparency`"
- the `java.awt.Paint` interface listed under "All Implemented Interfaces:"

A.5.6 javax.tv.media.MediaSelectControl

A.5.6.1 addMediaSelectListener

The following specification addition is considered to be present

If the specified listener is currently subscribed, no action is performed.

A.5.7 javax.tv.graphics.TVContainer

A.5.7.1 getRootContainer

The returns clause of this method is considered to be prefixed with the following:

On the first occasion this method is called for a particular xlet,

The returns clause is considered to have the following text added to it:

On subsequent calls to this method for a particular xlet, the implementation shall return the container without modifying it.

A.6 DAVIC 1.4.1p9 [3]

A.6.1 org.davic.mpeg

A.6.1.1 General

The following classes are considered to have a no argument protected constructor:

- ElementaryStream
- Service
- TransportStream

A.6.1.2 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 Chapter 9, Application Format

A.6.2.1 Section 9.4.7. "The MPEG-2 Section Filter API"

In the description of this class there are 2 instances of a cross reference to DAVIC part 10, section 115.3. In each case this shall be considered as a reference to DAVIC part 10, section 12.5.3.

A.6.3 org.davic.mpeg.dvb

A.6.3.1 General

The following classes are considered to have a no argument protected constructor:

- DvbService
- DvbElementaryStream
- DvbTransportStream

A.6.4 org.davic.mpeg.sections

A.6.4.1 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.6.4.2 Section

A.6.4.2.1 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.6.4.2.2 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.6.4.2.3 getFullStatus()

Is considered to have the following text appended to its description:

Returns true when the Section object contains valid data.

A.6.4.3 SectionFilter

A.6.4.3.1 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.6.4.3.2 startFiltering(all signatures)

In these methods, the description of when the FilterResourceException shall be thrown is considered to have the following text appended to its description:

Note that this is applied whether the parent section filter group is connected to a TS or not.

A.6.4.3.3 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.6.4.3.4 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.6.4.3.5 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.6.4.4 SectionFilterGroup

A.6.4.4.1 attach

A `NotAuthorizedException` is added to the definition of this method.

A.6.4.4.2 Constructors

The constructors are considered to have.

Throws `IllegalArgumentException` if `numberOfFilters < 1`.

A.6.4.4.3 sectionSize

All methods with a `sectionSize` parameter are considered to have the following text appended to their descriptions:

Throws `IllegalArgumentException` if `sectionSize < 1`.

A.6.4.4.4 newRingSectionFilter

Is considered to have the following text appended to its description:

Throws `IllegalArgumentException` if `ringSize < 1`.

A.6.4.4.5 Constructor(int, boolean)

Is considered to have the following text appended to its description:

The scope of the `resourcePriority` shall be a single application only.

A.6.4.5 TimeOutEvent

The specification is considered to include `org.davic.mpeg.sections.TimeOutEvent` as specified below:

org.davic.mpeg.sections TimeOutEvent

Declaration

```
public class TimeOutEvent extends org.davic.mpeg.sections.EndOfFilteringEvent
```

```
java.lang.Object
|
+-- java.util.EventObject
    |
    +-- org.davic.mpeg.sections.org.davic.mpeg.sections.SectionFilterEvent
        |
        +-- org.davic.mpeg.sections.org.davic.mpeg.sections.EndOfFilteringEvent
            |
            +-- org.davic.mpeg.sections.TimeOutEvent
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

This event is generated if section filter operations time out within the period specified by the `setTimeout()` method. For a `SimpleSectionFilter` it will be generated if no sections arrive within the specified period. For a `TableSectionFilter`, it will be generated if the complete table does not arrive within the specified time. For a `RingSectionFilter`, it will be generated if the specified time has elapsed since the arrival of the last section being successfully filtered.

Constructors

TimeOutEvent(SectionFilter, Object)

```
public TimeOutEvent(SectionFilter f, java.lang.Object appData)
```

This constructs an `TimeOutEvent` event for the specified `SectionFilter` object.

Parameters:

`f` - the `SectionFilter` object which timed out

`appData` - application data that was passed to the `startFiltering` method

Methods

getSource()

```
public java.lang.Object getSource()
```

This returns the `SectionFilter` object which timed out

Overrides:

`EndOfFilteringEvent.getSource()` in class `EndOfFilteringEvent`

A.6.5 org.davic.media

A.6.5.1 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. If the player is not started then the exception shall be thrown.

A.6.5.2 MediaTimePositionChangedEvent

Add the following constructor:

```
MediaTimePositionChangedEvent (
    Controller from,
    int previous,
    int current,
    int target,
    Time mediaTime)
```

With the following definition of parameters:

Parameters:

- from - the controller whose media position was changed
- previous - the state the controller was in before this event
- current - the state the controller was in at the time the event was generated
- target - the state that the controller is heading to
- mediaTime - the media time after the change

A.6.5.3 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( org.davic.mpeg.ElementaryStream[], int[], int[] )
NotAuthorizedMediaException(Service, int major_reason,int minor_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. The exception has no detail message.

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. The exception has no detail message.

Parameters:

`s` - the service which could not be accessed

`reason` - the reason why the service could not be accessed

NotAuthorizedMediaException(ElementaryStream[], int[], int[])

```
NotAuthorizedMediaException(ElementaryStream[] e, int[] major_reason, int[] minor_reason)
```

Constructor for exception due to failure accessing one or more MPEG elementary streams. The caller of this constructor is responsible for ensuring the three 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

`major_reason` - the major reason why the exception was thrown for each elementary stream

`minor_reason` - the minor reason why the exception was thrown for each elementary stream

Use of the constructor `NotAuthorizedMediaException(ElementaryStream[] e, int[] reason)` will result in the major reason for each elementary stream being the one specified in the reason parameter to the method and the minor reason being `OTHER` as defined in `NotAuthorizedInterface`.

NotAuthorizedMediaException(Service, int, int)

```
public NotAuthorizedMediaException(org.davic.mpeg.Service s, int major_reason, int minor_reason)
```

Constructor for exception due to failure accessing an MPEG service

Parameters:

`s` - the service which could not be accessed

`major_reason` - the major reason why the service could not be accessed

`minor_reason` - the minor reason why the service could not be accessed

Since:

MHP 1.0.2

A.6.5.4 LanguageControl

The following description of semantics is considered to be present:

If more than one stream with the same language exists, the behaviour of `selectLanguage(String)` is to select the first listed in the network signalling.

NOTE: This is equivalent to item [b](#) under [11.4.2.3](#), "Default media player behaviour" on page 113.

A.6.6 org.davic.net

A.6.6.1 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()
```

Constructor without reason.

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.6.6.2 Locator**A.6.6.2.1 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.6.6.2.2 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.6.6.3 tuning**A.6.6.3.1 NetworkInterfaceController****A.6.6.3.1.1 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.6.6.3.1.2 `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.6.6.3.1.3 `tune()`

Replace "this `NetworkInterface`" with "the `NetworkInterface` reserved by this `NetworkInterfaceController`" in section H.5.4.3 of DAVIC specification.

A.6.6.3.2 `NetworkInterface`

A.6.6.3.2.1 Protected constructor

This class is considered to have a no argument protected constructor with the following statement attached to it:

This constructor is provided for the use of implementations and specifications which extend this specification. Applications shall not define sub-classes of this class. Implementations are not required to behave correctly if any such application defined sub-classes are used.

A.6.6.4 `ca`

A.6.6.4.1 `CAMessage`

The following class definition is considered to be a normative part of the specification:

```
public class CAMessage
    extends java.lang.Object
```

This class represents messages to CA modules.

Constructors

```
public CAMessage(byte [] data)
```

Constructor for the message

Parameters:

data - message data

Method Detail

getData

```
public byte[] getData()
```

Returns:

the data of the message

A.6.6.4.2 CAModule

A.6.6.4.2.1 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.6.6.4.2.2 isDescramblable(ElementaryStream streams[])

Is considered to have the following text appended to its description:

If an empty array is passed in, returns true.

A.6.6.4.2.3 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.

The description of this method is considered to have the following text added to it.

In systems based on the DVB common interface, messages sessions opened using this method shall be mapped onto the CA pipeline for the module represented by this CAModule instance as defined in section 6.8. of the common interface extensions specification. Neither the `module_id` or the `resource_id` of the module are visible to the application. It is the responsibility of the platform to perform the relevant mapping.

NOTE: The document referred to as "common interface extensions specification." is [TS 101 699 \[71\]](#).

The following text:

Throws: ModuleResourceNonExistentException

raised if the specified resource cannot be addressed. This includes situations where the resource is not present in the module as well as addressing what would be a public resource in a DAVIC CA0 / DVB-CI system.

Is considered to be replaced by:

Throws: ModuleResourceNonExistentException

raised if the specified resource is not present in the module.

NOTE: The `ModuleResourceVersionTooLowException` shall never be thrown in this version of this specification.

A.6.6.4.2.4 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.6.6.4.2.5 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.

NOTE: The document referred to as "common interface extensions specification." is [TS 101 699 \[71\]](#).

A.6.6.4.2.6 Protected constructor

This class is considered to have a no argument protected constructor with the following statement attached to it:

This constructor is provided for the use of implementations and specifications which extend this specification. Applications shall not define sub-classes of this class. Implementations are not required to behave correctly if any such application defined sub-classes are used.

A.6.6.4.2.7 Additional methods"

The following specification is considered to contain the following additional methods:

getApplicationTitle

```
/**
 * Retrieves the Application Title String.

This functionality is provided at low lowel (within MMI and Application Information
sessions defined in EN50221) by the message application_info() message (Application
Information resource, see 8.
4.2.2)

 * @exception ModuleUnavailableException raised if the physical CA module
 *         has been removed and is not available any more
 * @return the application title string
 */
public String getApplicationTitle() throws ModuleUnavailableException;
```

enterApplication

```
/**
 * Requests the module to enter start the application and enter
 * the main application menu.
```

This functionality is provided at low lowel (within MMI and Application Information

sessions defined in EN50221) by the message `enter_menu()` message (Application Information resource, see 8.4.2.3)

```
* @exception ModuleUnavailableException raised if the physical CA module
*         has been removed and is not available any more
*/
public void enterApplication() throws ModuleUnavailableException;
```

closeMMI

```
/**
 * Requests the module to leave
 * the complete tree of the current high-level MMI dialogs.
 *
 * This functionality is provided at low level (within MMI and Application Information
 * sessions defined in EN50221) by the message close_mmi() message (MMI resource, see 8.6.2.1)
 *
 * @exception ModuleUnavailableException raised if the physical CA module
 *         has been removed and is not available any more
 */
public void closeMMI() throws ModuleUnavailableException;
```

A.6.6.4.3 CAModuleManager

A.6.6.4.3.1 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.6.6.4.3.2 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.6.6.4.4 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.6.6.4.5 **MMIObjct**

The following class definition is considered to be a normative part of the specification:

```
org.davic.net.ca
MMIObjct
```

Syntax

```
public class MMIObjct
```

```
java.lang.Object
```

```
|
```

```
+--org.davic.net.ca.MMIObjct
```

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.6.6.4.6 DescramblerProxy

A.6.6.4.6.1 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.

The description of this method is considered to be extended by the following text:

In systems based on the DVB common interface this maps onto `ca_pmt` with `ca_pmt_cmd_id = ok_descrambling` (Common Interface specification, section 8.4.3.4). and the `NotAuthorizedException` shall never be thrown.

The following text is considered to be appended to the description:

In systems based on the DVB common interface, this maps onto `ca_pmt` with `ca_pmt_cmd_id = ok_descrambling` (Common Interface specification, section 8.4.3.4).

A.6.6.4.6.2 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.6.6.4.6.3 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.6.6.4.6.4 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.6.6.4.6.5 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.6.6.4.6.6 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.6.6.4.6.7 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 affects 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.6.6.4.6.8 `startDescramblingDialog`

The description of this method is considered to be extended by the following:

In systems based on the DVB common interface, the `NotAuthorizedException` shall never be thrown.

A.6.6.4.7 `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.6.6.4.8 `ModuleResponseEvent`

A.6.6.4.8.1 `Protected Constructor`

This class is considered to have a no argument protected constructor with the following statement attached to it:

This constructor is provided for the use of implementations and specifications which extend this specification. Applications shall not define sub-classes of this class. Implementations are not required to behave correctly if any such application defined sub-classes are used.

A.6.6.4.9 `NewModuleEvent`

The class description for this class is considered to have the following text appended to it:

In the case of a CA module based on the DVB common interface, this event shall only be generated when module initialisation has been completed. Hence an application has no means to detect when a module is inserted but has not yet been initialised.

A.6.6.4.10 `PIDChangeEvent`

The following the text:

This event is generated as part of the Host Control functionality in the Common Interface / CA0 to signal that an elementary stream should be substituted with an other one. This event is intended for data services only. Television services that are presented using a high level media player API (whose implementation implicitly handles descrambling) will also have this event handled implicitly by that implementation.

Is considered to be replaced with the following:

In systems based upon the DVB Common Interface this event is generated in response to the Host Control `replace` / `clear_replace` requests.

NOTE: This event is for information only. The platform is responsible for implementing the requests from the CA system. See also [R206 \[80\]](#).

This class is considered to inherit from `org.davic.net.ca.CAEvent` instead of `org.davic.net.ca.DescramblerEvent`.

The text:

```
public PIDChangeEvent(short oldPid,
                    short newPid,
                    Object descramblerProxy)
```

and:

`descramblerProxy` - the `DescramblerProxy` object representing the descrambling resource which is the source of the event.

is considered to be replaced with:

```
public PIDChangeEvent(short oldPid,
                    short newPid,
                    Object caModule)
```

and:

caModule - the CAModule object representing the CA system which is the source of the event.

The text:

```
public Object getSource()
Returns the DescramblerProxy that is the source of the event.
```

Overrides:

getSource in class DescramblerEvent

is considered to be replaced with:

```
public Object getSource()
Returns the CAModule that is the source of the event.
```

Overrides:

getSource in class CAEvent

A.6.6.4.11 TuneRequestEvent

The following text:

This event is generated as part of the Host Control functionality in the Common Interface / CA0.

Is considered to be replaced with the following:

In systems based upon the DVB Common Interface this event is generated in response to the Host Control tune request.

NOTE 1: This event is only guaranteed to be delivered to applications that survive the service selection caused by the Host Control tune request (see 11.6.4, "Conditional Access API" on page 125).

NOTE 2: This event is for information only. The platform is responsible for implementing the service selection autonomously in response to the request from the CA system.

This class is considered to inherit from `org.davic.net.ca.CAEvent` instead of `org.davic.net.ca.DescramblerEvent`.

The text:

```
public TuneRequestEvent(Locator locator,
                    Object descramblerProxy)
```

and:

descramblerProxy - the DescramblerProxy object representing descrambler resource which is the source of the event

is considered to be replaced with:

```
public TuneRequestEvent(Locator locator,
                    Object caModule)
```

and:

caModule - the CAModule object representing the CA system which is the source of the event.

The text:

```
public Object getSource()
```

Returns the DescramblerProxy that is the source of the event.

Overrides:

getSource in class DescramblerEvent

is considered to be replaced with:

```
public Object getSource()
```

Returns the CAModule that is the source of the event.

Overrides:

getSource in class CAEvent

A.6.6.5 `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

A.6.6.6 `dvb.DvbLocator`

A.6.6.6.1 `DvbLocator(int, int, int, int, int[])`

In the constructor `DvbLocator(int, int, int, int, int[])`, the name of the last parameter in the method signature shall be considered to be "componenttags".

A.6.6.6.2 Additional method

The following method is considered to be present:

```
/** Returns the textual service identifier, if one was provided to the constructor.
 * @return the textual service identifier, null if not present
 * @since MHP1.0.1
 */
public String getTextualServiceIdentifier() ;
```

A.6.6.6.3 `getOriginalNetworkId`

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.6.7 `org.davic.net.tuning`

A.6.7.1 Figure H-1

In Figure H-1: " Tuning API object model: Base classes. " in the class `NetworkInterface`, the method `getURL()` shall be considered to be "`getLocator()`"

A.6.7.2 Figure H-2

In:

Figure H-2: " Tuning API object model: Exceptions."

the exception `IncorrectURLException` shall be considered to be absent.

A.6.8 Extensibility and Over-Riding

The DAVIC specification [DAVIC 1.4.1p9 \[3\]](#) is considered to include the following;

The addition of public or protected constructors, methods or fields to the `org.davic` packages is allowed except where this would cause a compliant MHP application to fail. Examples of the latter include the adding of abstract methods to classes or adding of methods to interfaces which application classes will implement.

Within the `org.davic` package and its sub packages, overriding inherited public and protected methods not specified as being overridden is an allowable implementation option.

NOTE: This language does not apply to constructors or to fields, because inherited fields and constructors cannot be overridden in the Java language.

A.7 HAVi [50]

A.7.1 Drafting conventions

This specification does not follow the ETSI drafting rules as specified in [TR 101 262 \[82\]](#). In particular, the term "should" shall be interpreted in terms of the ETSI drafting rules as being closer to "shall" than "should".

A.7.2 General

A.7.2.1 Thread-safety

As with the `java.awt` package, the `org.havi.ui` package is not required to be thread safe.

A.7.3 Event mechanism

This section presents a clarification for the HAVi event mechanism.

A.7.3.1 Introduction

One of the reasons for the HAVi event mechanism is to provide a mechanism for widgets to request a type of input that may not be available on the input device supported by a platform. A HAVi widget can indicate its input preference by implementing one or more `HxxxInputPreferred` interfaces. This is important to enable application provided widgets to have access to the same capabilities as have always been available to platform provided ones.

An `HxxxInputPreferred` interface indicates the widget that implements the interface expects a certain type of input events. E.g. on systems with limited input devices the platform can supply a virtual keyboard to generate the interface-specific events.

The `Hxxxable` and `HxxxValue` interfaces group methods common to various `HVisible` subclasses; they do not imply additional support from the platform.

The `HxxxInputPreferred` interfaces all imply possible support by a (virtual) keyboard. The platform shall provide a way for the user to generate the input expected by the focused component. This could mean automatically presenting some kind of virtual keyboard when a focused component requires input that could not be generated in another way. The platform can determine which virtual keys a widget requires by looking at the `HxxxInputPreferred` interfaces it implements.

For `keyCode` input, `HNavigationInputPreferred.getNavigationkeys()` and `HKeyboardInputPreferred.getValidInput()` allow the platform to determine which `keyCodes` the particular widgets expect. Platform implementations shall provide the user with a means to generate the `keyCodes` expected by all `HxxxInputPreferred` interfaces implemented by the widget having input focus (with the exception of `keyCodes` not supported according to `org.havi.ui.event.HKeyCapabilities`).

HAVi events generated by the platform and sent to a `HComponent` are `HActionEvent`, `HAdjustmentEvent`, `HItemEvent`, `HTextEvent`, `HRcEvent`, `HFocusEvent` and `HKeyEvent`.

A.7.3.2 Overview of HAVi events

`HComponents` indicate they want to receive `HFocusEvents` by implementing the `HNavigationInputPreferred` interface. For `HNavigationInputPreferred`, the `keyCodes` for navigation keystrokes generated on the `HNavigationInputPreferred` will be passed to the `HNavigationInputPreferred` as an `HFocusEvent` `transferId`. The `HNavigationInputPreferred` will process an `HFocusEvent` of id `FOCUS_TRANSFER` in its `processHFocusEvent` method by calling `requestFocus` on the `HNavigable` associated with the `keyCode` returned by `HFocusEvent`. `getTransferId()`, if there is one.

Implementations of `HNavigationInputPreferred` will not send this `HFocusEvent` to `FocusListeners` or `HFocusListeners`, it is only used by the platform to communicate the `transferId` to the `HNavigationInputPreferred`.

The manner in which the keystrokes are generated is implementation-specific, as some platforms may supply a more extensive hardware keyboard, while other platforms may choose to provide a virtual keyboard to generate these keystrokes.

Also, HAVi allows generation of the HAVi events on top of AWT (in `HComponent.processEvent`) from other events as an implementation option, to allow for platforms unable to generate HAVi event directly. As a result of this, it is platform specific whether `processEvent` is ever called with a HAVi event as argument, or that the event is generated in this method from other events. In such an implementation, a widget may receive the original, unexpected, platform-specific, non-HAVi `java.awt` events as well as the translated HAVi events. To avoid confusion, a HAVi widget is supposed to process only the HAVi events and applications should ignore other events (i.e. not add listeners for `java.awt` events and not use them through `processXxxEvent` methods).

For the handling of navigation keys, an implementation of the `HNavigationInputPreferred` interface receives the `keycodes` for navigation as the `transferId` of `org.havi.ui.event.HFocusEvent`, and not as a `java.awt.event.KeyEvent` or `org.havi.ui.event.HKeyEvent`. (On platforms which generate HAVi events from other events, it is possible that the component does receive a `java.awt.event.KeyEvent`, however this is platform-specific and applications should not rely on this).

`HComponent.processEvent` is responsible to dispatch `HFocusEvents` to the `processHFocusEvent` method; this method will call `requestFocus` on the target of a `transferId` for a `FOCUS_TRANSFER` event, and send `HFocusEvents` of type `FOCUS_GAINED` and `FOCUS_LOST` to any registered `HFocusListeners` for classes that support adding these listeners. Depending on which other HAVi interfaces the component implements and on whether it extends `HVisible`, this method will also change the state of the widget, play appropriate sounds, and notify any listeners specified in these other interfaces for a focus change.

An `HComponent` implementing `HKeyboardInputPreferred` will receive input in the form of `HKeyEvents`. On platforms with limited input devices, the platform can present a virtual keyboard for the user to generate the requested `keyCodes`. The platform can use the input type of the `HKeyboardInputPreferred` to determine which keys to present on this virtual keyboard, however, the platform is not required to prevent the user from generating more `keyCodes` than those requested by the `HKeyboardInputPreferred`.

HAVi allows to generate the `HKeyEvents` from other events in `HComponent.processEvent`, so any `KeyEvents` received by the component which are not `HKeyEvents` should be ignored as platform-specific.

`HComponent.processEvent` is responsible for dispatching `HKeyEvents` to the `processHKeyEvent` method; this method will send the `HKeyEvent` to any registered `HKeyListener`s for classes that support adding these listeners.

If an `HComponent` implements both `HNavigationInputPreferred` and `HKeyboardInputPreferred`, the platform will provide a way for the user to generate the `keyCodes` for both `HFocusEvents` and `HKeyEvents` in an unambiguous way (e.g. while in edit mode, `keyCodes` are converted into `HKeyEvents`, otherwise they generate `HFocusEvents`). The `getEditMode` method can be called by the platform to determine the current input mode of the widget. The platform will provide a platform specific way to switch between any different input modes as required by any limitations of its input devices, and notify the widget of this change by sending it an appropriate `START_CHANGE` or `END_CHANGE` event.

If the same `keyCode` is expected for different input types (e.g. `VK_ENTER` is set as navigation code while the widget also expects it as an `HKeyEvent`), then the platform will present the user with means to separately generate each of the HAVi events for that `keyCode` (in the case of the example, an `HFocusEvent` with `transferId VK_ENTER` and an `HKeyEvent`). Application authors should realize that to use `keyCodes` which are likely to have multiple functions may force some platforms to present a less than perfectly user-friendly display.

For platforms that generate HAVi events in `HComponent.processEvent`, extra events may be received in addition to the HAVi events. For example, for `HFocusEvent FOCUS_GAINED` and `FOCUS_LOST`, if the `HFocusEvent` is generated directly, `processEvent` should send this `HFocusEvent` to both `processHFocusEvent` and to `processFocusEvent` (the second one will happen in the superclass method). If the platform generates HAVi events in `processEvent`, when a `FocusEvent` is received by `HComponent.processEvent`, it will be sent to `processFocusEvent` and a new `HFocusEvent` will be created to send to `processHFocusEvent`.

For the other HAVi events, like `HActionEvent`, `HAdjustmentEvent`, `HItemEvent` and `HTextEvent` events, the behaviour is analogous. The platform will provide a way for the user to generate the events, either directly or from other events (translated in `HComponent.processEvent`), on `HComponents` that implement the `HActionInputPreferred`, `HAdjustmentInputPreferred`, `HSelectionInputPreferred`, and `HKeyboardInputPreferred` interfaces, respectively.

Implementations of the interfaces do not have to override `processEvent` so as to send these events to the appropriate `processHxxEvent` method, this will be the responsibility of `HComponent.processEvent`. The `processHxxEvent` method will call any registered listeners and perform any other specified actions, such as changing states, playing sounds, etc.

`HAdjustmentInputPreferred` and `HSelectionInputPreferred`, which extend `HOrientable`, allow the platform to select a way for the user to generate `HAdjustmentEvents` and `HItemEvents` which best matches the component's representation.

A.7.3.3 Relation between HAVi events and AWT events

Applications which extend from `java.awt.Component` (and not `org.havi.ui.HComponent`) and which register event listeners from the `java.awt.event` package shall receive the behaviour defined for the `java.awt.event` package. If events from the `org.havi.ui.event` package are generated on such a `Component`, these shall be sent to any registered listeners according to the behaviour defined for their parent class in `java.awt.event`.

HAVi allows implementations that generate `HxxxEvents` from other events, so an `HComponent` may receive the `HxxxEvent` in addition to those other events. For example, you could get both `havi.ui.event.HKeyEvents` sent to `HKeyListeners` and `awt.event.KeyEvents` sent to AWT `KeyListeners`.

For subclasses that extend from `Component` and not from `HComponent`, it is implementation dependent whether they receive the events defined in `awt.event` or a subclass of these events defined in `havi` (e.g. `org.havi.ui.event.HKeyEvent` instead of `java.awt.event.KeyEvent`). However, in the latter case the behaviour will be as defined for the parent class, e.g. an `org.havi.ui.event.HKeyEvent` will go to `KeyListeners` added with `Component.addKeyListener`. Also, applications cannot assume that such `Components` will be able to receive `KeyEvents` with `keyCodes` other than those specified in the MHP minimum profile.

A.7.3.4 Application guidelines

Any implementation of a `HxxxInputPreferred` interface shall implement the processing of the `HxxxEvent` and calling of listeners in the `processHxxEvent` method (where this event processing is not inherited from a parent class). It is the platforms' responsibility to provide a way in which the user can generate the events requested by the widget. Since the HAVi events may be generated in the `processEvent` method on some platforms, interface implementers should not assume that the `processEvent` method will receive the HAVi events on every platform. On platforms where the HAVi events are generated in `processEvent`, it is the platform's responsibility to call the `processHxxEvent` method with the HAVi event.

A.7.4 org.havi.ui

A.7.4.1 HActionable

The class `HListGroup` is considered to be removed from the list of "Platform Classes".

A.7.4.1.1 `getActionSound`

The description of this method shall be considered to be replaced with the following

Return the last action sound set by the `setActionSound` method or null if no action sound has been set.

A.7.4.2 `HActionInputPreferred`

The description of this method shall be considered to have the following text added:

If this `HActionInputPreferred` has no action command then an empty string shall be returned.

A.7.4.3 `HAdjustmentInputPreferred`

A.7.4.3.1 Interface description

In the interface description of `HAdjustmentInputPreferred` after the sentence:

For platforms with a restricted number of physical keys this may involve a "virtual keyboard" or similar mechanism.

is considered to be extended by the following text:

The system might use the information returned by the method `getOrientation()` of the super interface to select appropriate key mappings for this event. The mechanisms to generate this event shall not be effective while the component is disabled (see `HComponent.setEnabled()`).

Also, the following text:

All interoperable implementations of the `HAdjustmentInputPreferred` interface must extend `HComponent`.

Is considered to be replaced with:

Widgets of HAVi compliant applications implementing the `HAdjustmentInputPreferred` interface must have `HComponent` in their inheritance tree.

Also, in the following text:

Note that the `java.awt.Component` method `isFocusTraversable` should always return true for a `java.awt.Component` implementing this interface.

the word "should" is considered to be replaced with "shall".

A.7.4.3.2 `getAdjustMode`

The last sentence of the description of `getAdjustMode()` is considered to be extended with:

Note that these events are ignored, if the component is disabled. See also: `HComponent.setEnabled()`.

A.7.4.3.3 `processHAdjustmentEvent`

The description of `processHAdjustmentEvent()` is considered to be extended with:

Widgets implementing this interface shall ignore `HAdjustmentEvents`, while the component is disabled. See also: `HComponent.setEnabled()`.

A.7.4.3.4 `setAdjustMode`

The description of `setAdjustMode()` is considered to be extended with:

Calls to this method shall be ignored, if the component is disabled. See also: `HComponent.setEnabled()`.

A.7.4.4 `HAdjustmentValue`

A.7.4.4.1 Class description

The class `HListGroup` is considered to be removed from the list of "Platform Classes".

A.7.4.4.2 `getAdjustmentSound`

This method is considered to have the following text added to the method description.

null shall be returned if this method is called before its corresponding set method.

A.7.4.4.3 `getBlockIncrement`

This method is considered to have the following text added to the method description.

1 shall be returned if this method is called before its corresponding set method.

A.7.4.4.4 `getUnitIncrement`

This method is considered to have the following text added to the method description.

1 shall be returned if this method is called before its corresponding set method.

A.7.4.5 `HAnimateEffect`

A.7.4.5.1 `getRepeatCount`

This method shall be considered to have the following text added to the method description.

Except for `HAnimateEffect` implementations that specify a different default, `getRepeatCount()` returns `REPEAT_INFINITE` if no call to `setRepeatCount()` has previously been made.

A.7.4.6 `HBackgroundDevice`

In the `setBackgroundConfiguration` method, the sentence below shall be considered to be added to the description of this method:

Applications can prevent or limit changes to configurations of other, not intended, devices by using constants `ZERO_GRAPHICS_IMPACT`, `ZERO_VIDEO_IMPACT` and `ZERO_BACKGROUND_IMPACT` in their configuration templates.

A.7.4.7 `HBackgroundImage`

A.7.4.7.1 Constructors - `HBackgroundImage(String)`, `HBackgroundImage(URL)`

The sentence:

Loading of the data for the object is not required at this time.

shall be considered to be replaced with:

Loading of the data for the object shall not happen at this time.

A.7.4.7.2 `load`

The following text is considered to be added to the description of this method:

Multiple calls to `load` shall each add an extra listener, all of which are informed when the loading is completed. If `load` is called with the same listener more than once, the listener shall then receive multiple copies of a single event.

A.7.4.8 `HComponent`

A.7.4.8.1 `processEvent`

The following text is a clarification of the HAVi specification:

The implementation of the method `HComponent.processEvent()` shall ensure that key events which are translated to HAVi events shall not be reported to `processKeyEvent()` or reported to `KeyListeners`. Key events which are not translated to HAVi events shall be reported to `processKeyEvent()` and `KeyListeners` as defined in the Java specification.

NOTE: If applications override `processEvent` they may terminally disturb these processes. Applications should not do this without extreme care, as the results may be very implementation dependent.

A.7.4.9 HComponentOrdering

A.7.4.9.1 `addAfter, addBefore`

In the returns clause, the text:

`is successfully added.`

shall be considered to read

`is successfully added or was already present.`

A.7.4.10 HEventManager

A.7.4.10.1 Class description

The following text is considered to be added to the description of this class:

The `HEventManager` class is intended to assist platform or subclass implementers with the handling of HAVi events. Implementations are not required to use this class to dispatch HAVi events. Applications should not extend the `HEventManager` class and implementations are not required to behave correctly if an application does extend this class. If an extended multicaster is desired, `AWTEEventManager` should be used rather than `HEventManager`.

Also, extend the following paragraph in the class description:

It is an implementation option for this class to insert other classes in the inheritance tree (for example `java.awt.AWTEEventManager`). It is allowed that this may result in `HEventManager` inheriting additional methods beyond those specified here.

with:

If this class does extend `java.awt.AWTEEventManager`, it is allowed for the fields defined in this class to be inherited from that parent class.

A.7.4.10.2 Constructor

The following text is considered to form part of the constructor description:

The parameters `a` & `b` passed to the constructor shall be used to populate the fields `a` & `b` of the instance.

A.7.4.10.3 `remove`

In the description of this method, replace:

`returns the resulting multicast listener`

with

`returns the result`

A.7.4.11 HFontCapabilities

A.7.4.11.1 `getSupportedCharacterRanges`

The returns clause of this method shall be considered to be extended with the following text:

`including where the capabilities of the font are unknown.`

A.7.4.12 HGraphicsDevice

A.7.4.12.1 `getBestConfiguration`

The following text shall be considered to be added to the end of the second paragraph of the method description:

If there are such equally best configurations, the one which is returned by this method is an implementation dependent selection from among those which are equally best.

A.7.4.12.2 `setGraphicsConfiguration`

In the `setGraphicsConfiguration` method, the sentence below:

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.

shall be considered to be extended as follows:

Applications can prevent or limit changes to configurations of other, not intended, devices by using constants `ZERO_GRAPHICS_IMPACT`, `ZERO_VIDEO_IMPACT` and `ZERO_BACKGROUND_IMPACT` in their configuration templates.

A.7.4.13 `HGraphicsConfigTemplate`

A.7.4.13.1 `setPreference`

The following text shall be added to the third paragraph of the method description:

Calling this method with null for the object parameter shall have no effect if the preference is not currently set in the template.

A.7.4.14 `HListElement`

A.7.4.14.1 Class description

The class description is considered to be extended by the following:

The methods `setIcon()` and `setLabel()` of `HListElement` shall not be used for elements, which are part of `HListGroup`. If an application requires to alter the content, it shall either replace the entire element, or remove it temporarily and re-add it after the content was changed.

A.7.4.15 `HListGroup`

A.7.4.15.1 Class description

In the following text:

The `HListGroup` is a user interface component representing a list of selectable items (`HListElements`) which contain static read-only graphical and / or textual content.

The phrase "static read-only" is considered to be removed.

Also, extend the following text:

Interoperable HAVi applications shall not add `HListElement` more than once. If an application requires items with identical contents (label and/or icon), then additional items shall be created. The behaviour of the `HListGroup` if duplicates are added is implementation specific.

With:

The methods `setIcon()` and `setLabel()` of `HListElement` shall not be used for elements, which are part of `HListGroup`. If an application requires to alter the content, it shall either replace the entire element, or remove it temporarily and re-add it after the content was changed.

Also, in the text:

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.

The phrase "if no elements are present" is considered to be replaced with:

if label and icon size are not set (see `HListGroupLook.getMinimumSize()`).

Also, under the parameters table under heading "Default parameter values exposed in the constructors" the description of the parameter "items" missing is considered to read:

The initial list of elements for this `HListGroup` or `null` for an empty list.

A.7.4.15.2 Fields

A.7.4.15.2.1 ITEM_NOT_FOUND

In the description:

A constant which may be returned from `getIndex` if the requested element is not found in the content.

The word "may" is considered to be replaced by "shall".

A.7.4.15.2.2 ADD_INDEX_END

In the description:

A constant for use with `addItem` and `addItems` which specifies that the new items should be appended to the end of the list.

The word "should" is considered to be replaced by "shall".

A.7.4.15.3 Use of "action" and "actioning"

The use of the words "action" and "actioning" in the following methods in `HListGroup` does not imply any connection between `HListGroup` and `HActionable`. There is no such connection in the API.:

- `getCurrentIndex()`
- `getCurrentItem()`
- `setCurrentItem()`
- `getSelectionIndices()`
- `getSelection()`

A.7.4.15.4 addItem(HListItem item, int index)

In the description of parameter "index" all occurrences of the word "items" is considered to be replaced by "item".

A.7.4.15.5 addItem(HListItem items[], int index)

In the parameter description:

item - the item to add.

is considered to be replaced by:

items - the items to add.

A.7.4.15.6 getIconSize

The description of this method is considered to be extended by:

If label and icon size do not match the size per element, the associated `HListGroupLook` is allowed to use other sizes during the rendering process. This size shall be used by `HListGroupLook` to calculate the size per element.

A.7.4.15.7 getLabelSize

The description of this method is considered to be extended by:

If label and icon size do not match the size per element, the associated `HListGroupLook` is allowed to use other sizes during the rendering process. This size shall be used by `HListGroupLook` to calculate the size per element.

A.7.4.15.8 getOrientation

The following text:

The orientation controls how an associated HLook lays out the component and affects the visual behaviour of the HAdjustmentEvent and HItemEvent events. HListGroups do not receive HAdjustmentEvents.

Is considered to be replaced by:

The orientation controls how an associated HLook lays out the component and affects the visual behaviour of HItemEvent events.

A.7.4.15.9 setFocusTraversal

In all cases the phrase "then null should be specified" is considered to be replaced by "then null shall be specified".

A.7.4.15.10 setItemSelected

The following text shall be considered to be added to the second sentence of the method description:

If the selection does not change then no HItemEvent shall be sent.

A.7.4.15.11 setIconSize

The description of this method is considered to be extended by:

If label and icon size do not match the size per element, the associated HListGroupLook is allowed to use other sizes during the rendering process. This size shall be used by HListGroupLook to calculate the size per element.

In the description of the size parameter, the following text:

Dimension(DEFAULT_ICON_SIZE, DEFAULT_ICON_SIZE)

is considered to be replaced by

Dimension(DEFAULT_ICON_WIDTH, DEFAULT_ICON_HEIGHT)

A.7.4.15.12 setLabelSize

The description of this method is considered to be extended by:

If label and icon size do not match the size per element, the associated HListGroupLook is allowed to use other sizes during the rendering process. This size shall be used by HListGroupLook to calculate the size per element.

A.7.4.15.13 setListContent

The sentence:

Any existing selection is discarded (which may cause an HItemEvent to be generated.)

Is considered to be replaced by

If any elements are selected, then the selection is discarded and an HItemEvent is generated.

A.7.4.15.14 setMove

In all cases the phrase "then null should be specified" is considered to be replaced by "then null shall be specified".

A.7.4.15.15 setScrollPosition

This method is considered to have the following text added:

Parameters

scroll - the scroll position

A.7.4.15.16 `setSelectionMode`

The parameter in the method signature called `adjust` shall be considered to be called `edit` as used in the parameter list.

A.7.4.16 `HListGroupLook`

A.7.4.16.1 `getMaximumSize`

The description of `getMaximumSize()` is considered to be replaced by the following:

Returns the size to present all elements of the specified `HVisible` plus any additional dimensions that the `HListGroupLook` requires for border decoration etc. If no elements are present, a dimension object is returned with width and height set to `java.lang.Short.MAX_VALUE`.

The extra space required for border decoration can be determined from the `getInsets()` and `getElementInsets()` methods. The behaviour is not defined for the case, when a subclass overrides these methods. Application developers shall not assume any influence on the returned dimensions.

The size per element shall be determined by calls to `getIconSize()` and `getLabelSize()` of `HListGroup`. If any of the values requests a default as specified by `DEFAULT_ICON_WIDTH`, `DEFAULT_ICON_HEIGHT`, `DEFAULT_LABEL_WIDTH` and `DEFAULT_LABEL_HEIGHT`, then an implementation specific default is used for the corresponding value(s).

Parameters:

`visible` `HVisible` to which this `HLook` is attached.

Returns:

A dimension object indicating this `HListGroupLook`'s maximum size. See also:

```
HListGroup.setIconSize()
HListGroup.setLabelSize()
HVisible.getMaximumSize()
```

A.7.4.16.2 `getMinimumSize`

The description of `getMinimumSize()` is considered to be replaced by the following:

Returns the size to present one element of the specified `HVisible` plus any additional dimensions that the `HListGroupLook` requires for border decoration etc.

The extra space required for border decoration can be determined from the `getInsets()` and `getElementInsets()` methods. The behaviour is not defined for the case, when a subclass overrides these methods. Application developers shall not assume any influence on the returned dimensions.

The size per element shall be determined by calls to `getIconSize()` and `getLabelSize()` of `HListGroup`. If any of the dimensions requests a default as specified by `DEFAULT_ICON_WIDTH`, `DEFAULT_ICON_HEIGHT`, `DEFAULT_LABEL_WIDTH` and `DEFAULT_LABEL_HEIGHT`, then an implementation specific default is used for the corresponding value(s).

Parameters:

`visible` `HVisible` to which this `HLook` is attached.

Returns:

A dimension object indicating this `HListGroupLook`'s minimum size. See also:

```
HListGroup.setIconSize()
HListGroup.setLabelSize()
HVisible.getMinimumSize()
```

A.7.4.16.3 `getPreferredSize`

The description of `getPreferredSize()` is considered to be replaced by the following:

Gets the preferred size of the `HVisible` component when drawn with this `HListGroupLook`.

If a default size for width and height was set with `HVisible.setDefaultSize()`, then the dimensions are rounded down to the nearest element (minimum of one) according to the orientation of the associated `HListGroup`, and any dimensions for border decorations etc. are added.

If no default size was set or only for one dimension (i.e. height is `NO_DEFAULT_HEIGHT` or width is `NO_DEFAULT_WIDTH`), then the unset dimension(s) shall be sufficiently large to present five elements according to the `HListGroup`'s orientation. Any dimensions for border decoration etc. are added.

The extra space required for border decoration can be determined from the `getInsets()` and `getElementInsets()` methods. The behaviour is not defined for the case, when a subclass overrides these methods. Application developers shall not assume any influence on the returned dimensions.

The size per element shall be determined by calls to `getIconSize()` and `getLabelSize()` of `HListGroup`. If any of the values requests a default as specified by `DEFAULT_ICON_WIDTH`, `DEFAULT_ICON_HEIGHT`, `DEFAULT_LABEL_WIDTH` and `DEFAULT_LABEL_HEIGHT`, then an implementation specific default is used for the corresponding value(s).

Parameters:

`visible` `HVisible` to which this `HListGroupLook` is attached.

Returns:

A dimension object indicating the preferred size of the `HVisible` when drawn with this `HListGroupLook`. See also:

```
HListGroup.setIconSize()
HListGroup.setLabelSize()
HVisible.getPreferredSize()
HVisible.setDefaultSize()
```

A.7.4.16.4 `getValue`

The first paragraph of `getValue()` is considered to be extended by the following:

A non-null value represents the scroll position of the associated `HListGroup`. The value shall never be less than zero.

And the description of returns is considered to be replaced by the following:

the non-negative scroll position associated with the specified pointer position or null

A.7.4.16.5 `hitTest`

In the description of `hitTest()` after

...then this method will return the index of that element.

is considered to be extended by the following:

The `HListGroup` shall interpret this index as current item. If the value is `ADJUST_THUMB`, then the caller shall use `getValue()` to retrieve the actual scroll position corresponding to the specified pointer position."

And the returns description considered to be extended by the following:

, or a non-negative element index.

A.7.4.16.6 `showLook`

The description of `showLook()` is considered to be replaced by the following:

This method is responsible for repainting the entire visible including the list content set on the `HListGroup`, and the visible background, subject to the clipping rectangle of the graphics object passed to it.

If the method modifies the clipping rectangle of the graphics object, it shall restore the original rectangle upon return.

`showLook()` paints the visible with its current background colour according to the `getBackgroundMode()` method of `HVisible` and draws any (implementation-specific) borders, regardless whether content is available or not. Note that by default the background mode is set to not paint a background. Furthermore on platforms which support transparent colours the background colour may be partially or completely transparent.

Any resources explicitly associated with this look shall be loaded by it during its creation. Note that the "standard" looks do not load content by default.

This method is called from the `paint()` method of `HVisible` and must never be called from elsewhere. Components wishing to redraw themselves shall call their repaint method in the usual way or call `widgetChanged()` under certain circumstances.

The labels of the associated `HListElements` shall be rendered by using the current text layout manager of the `HListGroup`. For each visible label is the `render()` method of `HTextLayoutManager` called. The position and size per label are specified as insets relatively to the bounds of `HListGroup`. Note that the bounds are independent of any borders of the `HListGroup`, but the insets have to include the borders per element, if any. The look shall use the method `getLabelSize()` of `HListGroup` to determine the size for each label. If the returned dimension object has `DEFAULT_LABEL_WIDTH` for the width and/or `DEFAULT_LABEL_HEIGHT` for the height as values, then this method shall use implementation specific value(s) as default(s) for the missing dimension(s) instead. If `getTextLayoutManager()` returns null, then labels shall not be rendered.

If supported, scaling of icons shall reflect the resize mode of the visible within the area of the respective list element. The look shall use the method `getIconSize()` of `HListGroup` to determine the size for each icon. If the returned dimension object has `DEFAULT_ICON_WIDTH` for the width and/or `DEFAULT_ICON_HEIGHT` for the height as values, then this method shall use implementation specific value(s) as default(s) for the missing dimension(s) instead.

Except for the alignment of labels and sizes of labels and icons, it is explicitly not defined, how this look arranges icons and labels within the elements' areas. Additionally, it is an implementation option to render labels and icons in other sizes than specified, if the available size per element is smaller or larger than label and icon size. It is also not defined, how the look presents the current item and selected items, or the current selection mode. The elements shall be layed out as specified by `getOrientation()` of the associated `HListGroup`.

When the associated `HListGroup` contains more elements than presentable, the look shall make the user aware of that condition, e.g. by displaying an additional scrollbar reflecting the current scroll position. Again, the visible means by which this information is conveyed is not defined and implementation dependent. It is an implementation option for `HListGroupLook` to draw elements before the scroll position, in order to fill the available space.

The behaviour of this method, when a subclass overrides the methods `getInsets()` or `getElementInsets()`, is not defined. Application developers shall not assume that the corresponding borders will appear as specified.

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 the default behaviour of `HListGroupLook` is to ignore this parameter, since the content of `HListGroup` is not state based. See also:

```
java.awt.Component.getBounds()
java.awt.Graphics.getClipBounds()
HListGroup.getCurrentItem()
HListGroup.getListContent()
HListGroup.getOrientation()
HListGroup.getScrollPosition()
HListGroup.setIconSize()
HListGroup.setLabelSize()
HListGroup.isItemSelected()
HTextLayoutManager.render()
```

A.7.4.17 HLook

A.7.4.17.1 General

In the description of HLook and all implementing classes, the term `clipRect` shall be interpreted to mean "clipping rectangle".

A.7.4.17.2 Class description

Only the first paragraph of the "Invocation Mechanism" section shall be considered to be present.

A.7.4.17.3 `getPreferredSize`

In the description of this method, the text:

"... then the return value is the size of the largest piece of content..."

shall be considered to be replaced with:

"...then the return value is a size that is sufficiently large to hold each piece of content..."

The equivalent replacement shall apply for `getMinimumSize()` and `getMaximumSize()`.

In HLook and all classes implementing this interface, the following text shall be considered to form part of the method description of `getPreferredSize()`:

If a default preferred size has been set for this HVisible (using `setDefaultSize(Dimension)`) and the default preferred size has an `NO_DEFAULT_WIDTH` then the return value is a `Dimension` with this height (obtained with `getDefaultSize()`) and the preferred width for the content plus any additional dimensions that the HLook requires for border decoration etc.

If a default preferred size has been set for this HVisible (using `setDefaultSize(Dimension)`) and the default preferred size has an `NO_DEFAULT_HEIGHT` then the return value is a `Dimension` with this width (obtained with `getDefaultSize()`) and the preferred height for the content plus any additional dimensions that the HLook requires for border decoration etc.

A.7.4.17.4 `showLook`

In `org.havi.ui.HLook.showLook()`, the text;

The `showLook` method should not modify the `clipRect` of the `Graphics` object that is passed to it.

Shall be clarified as follows:

The `showLook` method shall not modify the `clipRect` (clipping rectangle) of the `Graphics` object passed to it in a way which includes any area not part of that original `clipRect`. If any modifications are made, the original `clipRect` shall be restored.

A.7.4.18 HMultilineEntry

The following text shall be considered to be added to the class description:

A call to the inherited method `setDefaultLook(HSinglelineEntry)` shall behave the same as a call to `HSinglelineEntry.setDefaultLook(HSinglelineEntry)`.

A.7.4.19 HMultilineEntryLook

A.7.4.19.1 `getCaretCharPositionForLine`

The following text:

If an invalid line is specified an `IllegalArgumentException` is thrown If it cannot be moved the nearest position should be returned

shall be considered to be replaced by:

If an invalid line is specified an `IllegalArgumentException` is thrown. If the caret cannot be moved to the same column position on this line, the nearest position should be returned.

A.7.4.19.2 `getSoftLineBreakPositions`

The following text shall be considered to be added to the returns clause of this method

If there is no text content within the `HVisible`, a zero length array shall be returned.

A.7.4.19.3 `getVisibleSoftLineBreakPositions`

The following text is considered be added to the returns clause of this method

If there is no text content within the `HVisible`, a zero length array shall be returned.

A.7.4.20 `HNavigable`

A.7.4.20.1 Class description

In the following sentence:

Applications should assume that classes which implement `HNavigable` can only generate events of the type `HFocusEvent` in response to other types of input event.

The word "only" shall be considered to be not present.

A.7.4.21 `HOrientable`

A.7.4.21.1 Interface description

The following text:

All interoperable implementations of the `HOrientable` interface must extend `HComponent`.

Is considered to be replaced with:

Widgets of HAVi compliant applications implementing the `HOrientable` interface must have `HComponent` in their inheritance tree.

A.7.4.21.2 `getOrientation`

The following text:

The orientation controls how an associated `HLook` lays out the component and affects the visual behaviour of the `HAdjustmentEvent` and `HItemEvent` events. For example, the system might use this information to select appropriate key mappings for these events.

Is considered to be replaced with:

The orientation controls the layout of the component.

A.7.4.21.3 `setOrientation`

The following text:

The orientation controls how the associated `HLook` lays out the component.

Is considered to be replaced with:

The orientation controls the layout of the component.

A.7.4.21.3.1 Orientation constants

In the description of the orientation constants: `ORIENT_LEFT_TO_RIGHT`, `ORIENT_RIGHT_TO_LEFT`, `ORIENT_TOP_TO_BOTTOM` and `ORIENT_BOTTOM_TO_TOP` the word "shall" is considered to replace "should" in each instance of the phrase "should be rendered".

A.7.4.22 HScene

A.7.4.22.1 addAfter, addBefore

In the returns clause, the text:

is successfully added

shall be considered to read:

is successfully added or was already present.

A.7.4.22.2 getFocusOwner

The following text shall be considered to be added to the end of the method description:

if and only if this HScene is active.

A.7.4.23 HScreenConfigurationListener

The parameter "gce" for the report method shall be considered to have the following description.

gce - The event notifying the listener of the modification.

Add to errata to external references - HAVi

A.7.4.24 HSceneFactory

A.7.4.24.1 getDefaultHScene

The text:

... identical to calling `org.havi.ui.HScene.getDefaultHScene()`

shall be considered to read:

... identical to calling `org.havi.ui.HSceneFactory.getDefaultHScene()`

A.7.4.25 HSelectionInputPreferred

A.7.4.25.1 Interface description

The following text is considered to be added to the interface description:

The system must provide a means of generating `HItemEvent` events as necessary. For platforms with a restricted number of physical keys this may involve a "virtual keyboard" or similar mechanism. The system might use the information returned by the method `getOrientation()` of the super interface to select appropriate key mappings for this event. The mechanisms to generate this event shall not be effective while the component is disabled (see `HComponent.setEnabled()`).

Also, the text:

All interoperable implementations of the `HSelectionInputPreferred` interface must extend `HComponent`.

Is considered to be replaced with:

Widgets of HAVi compliant applications implementing the `HSelectionInputPreferred` interface must have `HComponent` in their inheritance tree.

Also, in the text:

Note that the `java.awt.Component` method `isFocusTraversable` should always return true for a `java.awt.Component` implementing this interface.

The word "should" is considered to be replaced with "shall".

A.7.4.25.2 `getSelectionMode`

The description of this method is considered to be extended by the following text:

Note that these events are ignored, if the component is disabled. See also: `HComponent.setEnabled()`.

A.7.4.25.3 `processHItemEvent`

The description of this method is considered to be extended by the following text:

Widgets implementing this interface shall ignore `HItemEvents`, while the component is disabled. See also: `HComponent.setEnabled()`.

A.7.4.25.4 `setSelectionMode`

The description of this method is considered to be extended by the following text:

Calls to this method shall be ignored, if the component is disabled. See also: `HComponent.setEnabled()`.

A.7.4.26 `HSingleLineEntry`

A.7.4.26.1 Class description

The term:

`TEXT_START_CHANGE` event

is considered to be replaced with:

an `HTextEvent` event with an id of `TEXT_START_CHANGE`.

A.7.4.27 `HStaticAnimation`

The following text is considered to be added to the description of this class:

Calling `setVisible(false)` on `HStaticAnimation` shall not automatically stop the animation hence applications are not required to call the `play()` method again when the animation again becomes visible. It is implementation dependent whether the animation continues from the last visible position or from where it would be if it kept running.

A.7.4.28 `HStaticRange`

A.7.4.28.1 Fields

A.7.4.28.1.1 `SCROLLBAR_BEHAVIOR`

The description of this field is considered to be replaced by the following text:

The `HStaticRange` shall behave as a scrollbar, i.e. the allowable values that may be set / returned for the `HStaticRange` shall be affected by the "thumb" offsets, and hence its value shall be able to vary between `[minimum + minThumbOffset, maximum - maxThumbOffset]`.

A.7.4.28.1.2 `SLIDER_BEHAVIOR`

The description of this field is considered to be replaced by the following text:

The `HStaticRange` shall behave as a slider, i.e. the allowable values that may be set / returned for the `HStaticRange` shall not be affected by the "thumb" offsets, and hence its value shall be able to vary between `[minimum, maximum]`.

A.7.4.28.2 `getOrientation`

The following text:

The orientation controls how an associated `HLook` lays out the component and affects the visual behaviour of the `HAdjustmentEvent` and `HItemEvent` events. For example, the system might use this information to select appropriate key mappings for these events.

Is considered to be replaced by:

The orientation controls how the associated HLook lays out the component.

A.7.4.28.3 `setBehavior`

The following text is considered to be appended to the method description:

If the new behaviour is `SCROLLBAR_BEHAVIOR` and the control's settings for range and thumb offsets are illegal, i.e. `minimum + thumbMinOffset` is equal or greater than `maximum - thumbMaxOffset`, then an `IllegalArgumentException` shall be thrown. If the control's value is not valid for the offsets, then the value shall be changed to the closest valid value.

A.7.4.28.4 `setRange`

The following text is considered to be appended to the method description:

If the maximum is greater than the minimum and the value of the control is outside the new range (subject to the control's current behaviour), then the value is changed to the closest valid value.

A.7.4.28.5 `setThumbOffsets`

The following text is considered to be appended to the method description:

If this control's behaviour is `SCROLLBAR_BEHAVIOR`, then the following rules apply: If the thumb offsets are illegal, i.e. `minimum + thumbMinOffset` is equal or greater than `maximum - thumbMaxOffset`, then an `IllegalArgumentException` shall be thrown. If the control's value is not valid for the specified offsets, then the value shall be changed to the closest valid value.

A.7.4.28.6 `setValue`

The method's parameter description is considered to be replaced by:

value - the value for this `HStaticRange`

Also, the following text is considered to be appended to the method description:

If the specified value is not valid, then the method shall round it to the closest valid value. An application can retrieve the corrected value by means of method `getValue()`.

A.7.4.29 `HVideoDevice`

A.7.4.29.1 `getBestConfiguration`

The following text is considered to be added to the end of the second paragraph of the descriptions of both methods of this name.

If there are such equally best configurations, the one which is returned by this method is an implementation dependent selection from among those which are equally best.

A.7.4.29.2 `setVideoConfiguration`

In the `setVideoConfiguration` method, the sentence below:

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.

shall be considered to be extended as follows:

Applications can prevent or limit changes to configurations of other, not intended, devices by using constants `ZERO_GRAPHICS_IMPACT`, `ZERO_VIDEO_IMPACT` and `ZERO_BACKGROUND_IMPACT` in their configuration templates.

A.7.4.30 `HVisible`

A.7.4.30.1 Class description

In the class description of `HVisible`, the sentence:

Most content is stored by reference, but text content is copied as `java.lang.String` objects are immutable. shall be considered to not be present.

A.7.4.30.2 Constructors

The following text shall be considered to form part of those constructor descriptions which have an `HLook` as a parameter:

Applications shall not use `HLOOKS` with this constructor unless those `HLOOKS` are specified as working with `HVisible`. If an `HLOOK` is used which is specified as only working with specific sub-classes of `HVisible` then the failure mode is implementation dependent.

A.7.4.30.3 `setDefaultSize`

The following text:

If this parameter or specifies a size smaller than an implementation-defined minimum size a `java.lang.IllegalArgumentException` will be thrown.

Shall be considered to be replaced by:

If this parameter specifies a size smaller than an implementation-defined minimum size, the preferred size of this component shall be set to that implementation-defined minimum size.

Also, the following text:

If the parent `Container` into which the `HVisible` is placed has no layout manager this method has no effect.

Is considered to be removed.

A.7.4.30.4 `setLookData`

The method description shall be considered to have the following text added.

If for this specified key a data object has been set, the old data object shall be replaced with the new one.

A.7.4.30.5 `setResizeMode`

The following text added shall be considered to be replaced:

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.

by:

Set the resize mode of this `HVisible`. If the associated `HLook` does not render content or if scaling is not supported, changing the mode may have no visible effect.

A.7.4.30.6 `setTextContent`

In the description of this method, the following sentence is considered to not be present:

Note that unlike `setGraphicContent`, `setAnimateContent` and `setContent`, the content is copied as it is not possible to store a reference to a `java.lang.String`.

A.7.5 `org.havi.ui.event`

A.7.5.1 `HActionEvent`

The text:

Modifiers are not used with the HAVi platform. Interoperable HAVi applications shall not use the return value of this method.

shall be considered to be replaced by:

Modifiers are not used for `HActionEvents` with the HAVi platform. Interoperable HAVi applications shall not use the return value of this method.

A.7.5.2 HItemEvent

A.7.5.2.1 Constructor

In the constructor for this class, the description of the item parameter is considered to be replaced by the following:

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`, `ITEM_CLEARED`, `ITEM_SET_CURRENT`, `ITEM_SET_NEXT` or `ITEM_SET_PREVIOUS`.

A.7.5.3 HKeyEvent

The following paragraph shall not apply in this specification:

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.

A.7.5.3.1 Constructors

The description of the 'id' parameter shall be considered to have the following text added:

This is the value that will be returned by the event object's `getID` method.

A.7.5.4 HRcEvent

The following changes shall be considered to apply in this class:

- Replace all occurrences of "action id" in the descriptions of the VK constants by "key code".
- Replace all occurrences of "event ids" in the descriptions of `RC_FIRST` and `RC_LAST` by "key codes". Tag both constants as being deprecated.
- Remove the following sentence from the class description: "Note that it is an implementation constraint that the `HrcEvent` event range should not intersect with the Java AWT key event range."
- Add to the parameter descriptions of "id" for both constructors: "in the range `KEY_FIRST` to `KEY_LAST`".

A.7.5.5 HRcCapabilities

In the method `isRepresentation(int aCode)`, all references to the parameter `id` shall be considered to be replaced with `keyCode`.

A.7.6 References to ISO/IEC10646-1:1993

All references to ISO/IEC10646-1:1993 are considered to be non-specific references to [ISO 10646-1 \[18\]](#). Also, the term "support for Unicode ranges" is considered to be "support for character ranges as specified by [ISO 10646-1 \[18\]](#)".

A.8 ISO/IEC 13818-6 [26]

A.8.1 Reconstruction of NPT

In equation 8-3, `SCR_Reference` is considered to be `STC_Reference`.

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 290	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 94).

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<N1; n++) { label_char } }</pre>	8	uimsbf	0x70	The label
<pre> descriptor_tag</pre>	8	uimsbf	N1	
<pre> label_char</pre>	8	uimsbf		

descriptor_tag: This 8 bit integer value with 0x70 identifies this descriptor.

label_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 298) 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	
<pre> descriptor_tag</pre>	8	uimsbf		
<pre> priority_value</pre>	8	uimsbf		
<pre> transparency_level</pre>	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 298.

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 Table B.21 on page 276 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	Table B.16, "BIOP::FileMessage syntax," on page 273
objectKind_data id_data kind_data	32 8 8	Table B.19, "BIOP::DirectoryMessage syntax," on page 274
objectKind_data	8	Table B.28, "BIOP::StreamMessage syntax," on page 281
objectKind_data eventName_data	32 8	Table B.30, "BIOP::StreamEventMessage syntax," on page 283
type_id_byte	32	Table B.21, "IOP::IOR syntax," on page 276
id_data kind_data	32 32	Table B.25, "Syntax of Lite Options Profile Body with ServiceLocation component.," on page 279

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: Content type descriptor	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 (note 1)
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
}				
}				
NOTE 1: If non-zero, this shall be the same as the content_length of the referenced FileMessage.				

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 shall be used to provide the media type mapping via table 4, "File type identification" on page 56. The extension portion of the filename shall not be used if this descriptor is provided.

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: Content type descriptor	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	32	uimsbf		
context_data_length	16	uimsbf	N4	
for (j=0; j<N4; j++) {				
context_data_byte	8	uimsbf	+	
}				
}				
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 276
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 BIOPProfileBody or LiteOptionsProfileBody .
				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	If BIOP_DELIVERY_PARA_USE is provided it shall be the first tap. 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).
<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<N4-18; n++) { additional_tap_byte } } for (n=0;n<N6;n++) { BIOP::LiteComponent{ componentId_tag component_data_length for (i=0; i<N7; i++) { component_data_byte } } } } </pre>	8	uimsbf		The MHP terminal may skip over the possible additional taps N6=N1-2
<pre> } </pre>				

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 Streams or in other Transport Streams. The following constraints are put on the use of the LiteOptionsProfileBody;

- LiteOptionsProfileBody references shall never be used in an IOR which is in the DSI referencing the service gateway.
- The target carousel is never mounted automatically by the implementation, but the application may do so using the MHP DSMCC API and where necessary, the tuning API.
- When the LiteOptionsProfileBody is encountered the application will get a ServiceXFRErrorEvent or a ServiceXFRException unless the object carousel which is the target of the lite options profile body reference is already mounted by the MHP terminal. In this latter case, the access to the object shall succeed unless the object is unavailable, e.g. the target file does not exist in the target carousel or the target carousel has lost its connection.

Table B.24 : Restrictions on the Lite Options Profile Body (Sheet 1 of 2)

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.

Table B.24 : Restrictions on the Lite Options Profile Body (Sheet 2 of 2)

Field	Restrictions	Source
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"
CosNaming::Name() {				pathName
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

Syntax	bits	Type	Value	Comment
DVBcarouselNSAPaddress {				
AFI	8	uimsbf	0x00	NSAP for private use
Type	8	uimsbf	0x00	Object carousel NSAP Address.
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	+	
service_id	16	uimsbf	+	(= MPEG-2 program_number)
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 aDescription_bytes in the DSM::Stream::Info_T structure and the possible other object info data following the structure. 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 ISO/IEC 13818-6 [26] . 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	+	
}				
objectKind_length	32	uimsbf	0x00000004	
objectKind_data	8	uimsbf	0x73747200	"str" type_id alias
objectInfo_length	16	uimsbf	N2	
DSM::Stream::Info_T {				objectInfo
aDescription_length	8	uimsbf	N3	aDescription
for (i=0; i<N3; i++) {				
aDescription_bytes	8	uimsbf	+	
}				
duration.aSeconds	32	simsbf	+	may be set to 0 to indicate undefined
duration.aMicroSeconds	32	uimsbf	+	may be set to 0 to indicate undefined
audio	8	uimsbf	+	
video	8	uimsbf	+	
data	8	uimsbf	+	
}				
for (i=0; i<N2-(N3+10); i++) {				
objectInfo_byte	8	uimsbf	+	
}				
serviceContextList_count	8	uimsbf	N4	serviceContextList

Table B.28 : BIOP::StreamMessage syntax (Sheet 2 of 2)

Syntax	bits	Type	Value	Comment
<pre> for (i=0; i<N4; i++) { context_id context_data_length for (j=0; j<N5; j++) { context_data_byte } } </pre>	32 16 8	uimsbf uimsbf uimsbf	N5 +	
<pre> messageBody_length taps_count for (i=0; i<N6; i++) { id use assocTag selector_length } } </pre>	32 8 16 16 16 8	uimsbf uimsbf uimsbf uimsbf uimsbf	* N6 (note 1) + + 0x00	<p>see B.2.4.4 "Timebases" see Table 4-12 in DVB Guidelines for Data Broadcasting</p> <p>no selector</p>
<p>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.</p>				

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 Table B.27 on page 281 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"> • 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 be at most one tap with use STR_NPT_USE, which MHP terminals shall interpret as described in ISO/IEC 13818-6 [26]. • 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). 	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 {		uimsbf		
aDescription_length	8	uimsbf	N3	aDescription
for (i=0; i<N3; i++) {				
aDescription_bytes	8	uimsbf	+	see BIOP StreamMessage
}				
duration.aSeconds	32	simsbf	+	see BIOP StreamMessage
duration.aMicroSeconds	32	uimsbf	+	see BIOP StreamMessage
audio	8	uimsbf	+	see BIOP StreamMessage
video	8	uimsbf	+	see BIOP StreamMessage

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 for (i=0; i<N4; i++) { eventName_length for (j=0; j<N5; j++) { eventName_data } } }	16 8 8	uimsbf uimsbf uimsbf	N4 N5 +	(including zero terminator)
for (i=0; i<N2 - (N3 + 14 + N4 + sum(N5)); i++) { objectInfo_byte }	8	uimsbf	+	
serviceContextList_count	8	uimsbf	N6	
for (i=0; i<N6; i++) { context_id context_data_length for (j=0; j<N7; j++) { context_data_byte } }	32 16 8	uimsbf uimsbf uimsbf	N7 +	
messageBody_length taps_count for (i=0; i<N8; i++) { id use assocTag selector_length }	32 8 16 16 16 8	uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf	* N8 (note 1) + + 0x00	see B.2.4.4 "Timebases" see Table 4-12 in DVB Guidelines for Data Broadcasting no selector
eventIds_count for (i=0; i<N4; i++) { eventId } }	8 16	uimsbf uimsbf	N4 +	(= eventNames_count)
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 288.

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 298) and version listener mechanisms (see DSMCCObject methods in annex P, "(normative): Broadcast Transport Protocol Access" on page 517) 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 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 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 288.

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.

With regard to postDiscontinuity indicator:

- it is optional for broadcasters to broadcast descriptors with this set to one however if they are broadcast then the following conditions shall be met
 - it shall be broadcast for at least 5 seconds before the underlying PCR discontinuity
 - the descriptors with both states of postDiscontinuity indicator shall be carried within the same DSMCC descriptor list section
 - within one second of the underlying PCR discontinuity, the NPTReferenceDescriptor must be updated to reflect the new underlying PCR
- it is optional for MHP terminals to take advantage of this signalling but terminals which do not take advantage of this must ignore descriptors with postDiscontinuity indicator set to 1.

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 287 and "Signalling of "do it now events"" on page 286.

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 287).

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 114 and 11.5.1, "Broadcast Transport Protocol Access API" on page 118

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 to the first instance of a "do it now" event detected under a particular combination of table id, table id extension & version number. Reception of subsequent copies of the particular event shall be ignored until a different version number is detected.

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 NPT reference descriptors
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 specification) - 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 currently signalled 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. The set of signalled timebases can change through time. When a timebase is not signalled then the behaviour of the MHP terminal shall be identical to that of the timebase being paused.

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, apart from the generation of the NPTRDiscontinuityEvent to registered listeners.

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. See also B.2.4.2.1.2, "Syntax" on page 285 on use of the postDiscontinuity.

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 applications have registered for 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 applications have registered for 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.

For each application, the MHP terminal is not required to monitor more than a single component delivering "do it now" stream events. So, if events from more than one DSM-CC StreamEventMessage are subscribed to no more than one stream component shall be specified as the source of StreamEventDescriptors carrying "do it now" events (i.e. the taps with use STR_EVENTUSE or STR_STATUS_AND_EVENT_USE shall have the same value when referring to "do it now" events).

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.

For each application, the MHP terminal is not required to monitor more than a single component delivering scheduled stream events. So, if events from more than one DSM-CC StreamEventMessage are subscribed to no more than one stream component shall be specified as the source of StreamEventDescriptors carrying scheduled events (i.e. the taps with use STR_EVENT_USE or STR_STATUS_AND_EVENT_USE shall have the same value when referring to scheduled events).

NOTE: Scheduled and "do-it-now" stream events can be carried on different stream components. The MHP terminal is required to be able to monitor one stream of each.

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

B.2.6 Informative Background

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.

B.2.7 DVB semantics of the transactionId field

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

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).
16-29	User-defined	Version	This shall be incremented every time the control message is updated. The value by which it is incremented should be one.
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.2.8 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 decompressed¹. 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.2.9 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.

Presence of the compressed_module_descriptor indicates that the data in the module has the "zlib" structure as defined in RFC 1950 [76].

Table B.33 shows the syntax of the compressed_module_descriptor:

Table B.33 : compressed_module_descriptor

	No. of bytes	Mnemonic	Value
compressed_module_descriptor(){			
descriptor_tag	1	uimsbf	0x09
descriptor_length	1	uimsbf	
compression_method	1	uimsbf	
original_size	4	uimsbf	
}			

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 [76]. Table B.34 shows the syntax of the ZLIB structure.

Table B.34 : zlib structure

	No. of bytes	Value	
zlib structure(){			
compression_method	1		
flags_check	1		
compressed_data	n		
check value	4		
}			

The MHP terminal shall support the Deflate compression algorithm as specified in RFC 1951 [77]. 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.2.10 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`.

In this specification the use of the `carousel_id_descriptor` for signalling is mandatory 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). 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 76.

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.2.10.1 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		
}			
}			

Table B.35 : Carousel identifier descriptor syntax

Syntax	bits	Type	Value
<pre> if(FormatID == 0x01) { ModuleVersion ModuleId BlockSize ModuleSize CompressionMethod OriginalSize TimeOut ObjectKeyLength for(i=0; i<N2; i++){ ObjectKeyData } for(i=0; i<N1-N2-21; i++){ private_data_byte } } </pre>	8	uimsbf	N2 ≤ 4
ModuleVersion	8	uimsbf	
ModuleId	16	uimsbf	
BlockSize	16	uimsbf	
ModuleSize	32	uimsbf	
CompressionMethod	8	uimsbf	
OriginalSize	32	uimsbf	
TimeOut	8	uimsbf	
ObjectKeyLength	8	uimsbf	
ObjectKeyData	8	bslbf	
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 [76] 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.2.10.2 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.2.11 Unavailability of a carousel

Broadcast carousels become permanently unavailable 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.2.12 Delivery of Carousel within multiple services

A single Object Carousel may be transmitted within more than one service. An example is a single multiplex containing multiple services from the same broadcaster. A common data service could be supplied in all services by including references to the carousel in all PMTs.

Carousels shall be considered identical if, in the PMTs of the services, all the following hold:

- Both services are delivered within the same transport stream.
- Both services list the boot component of the carousel on the same PID.
- The carousel_id_descriptor for the carousel are identical in both services (so the carousels have the same carousel Id and boot parameters).
- All association tags used in the carousel map to the same PIDs in both services.

In this case the carousel is transmitted over a single path, but the services are allowed to reference the carousel via a number of routes, including deferral to a second PMT via deferred association tags.

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:

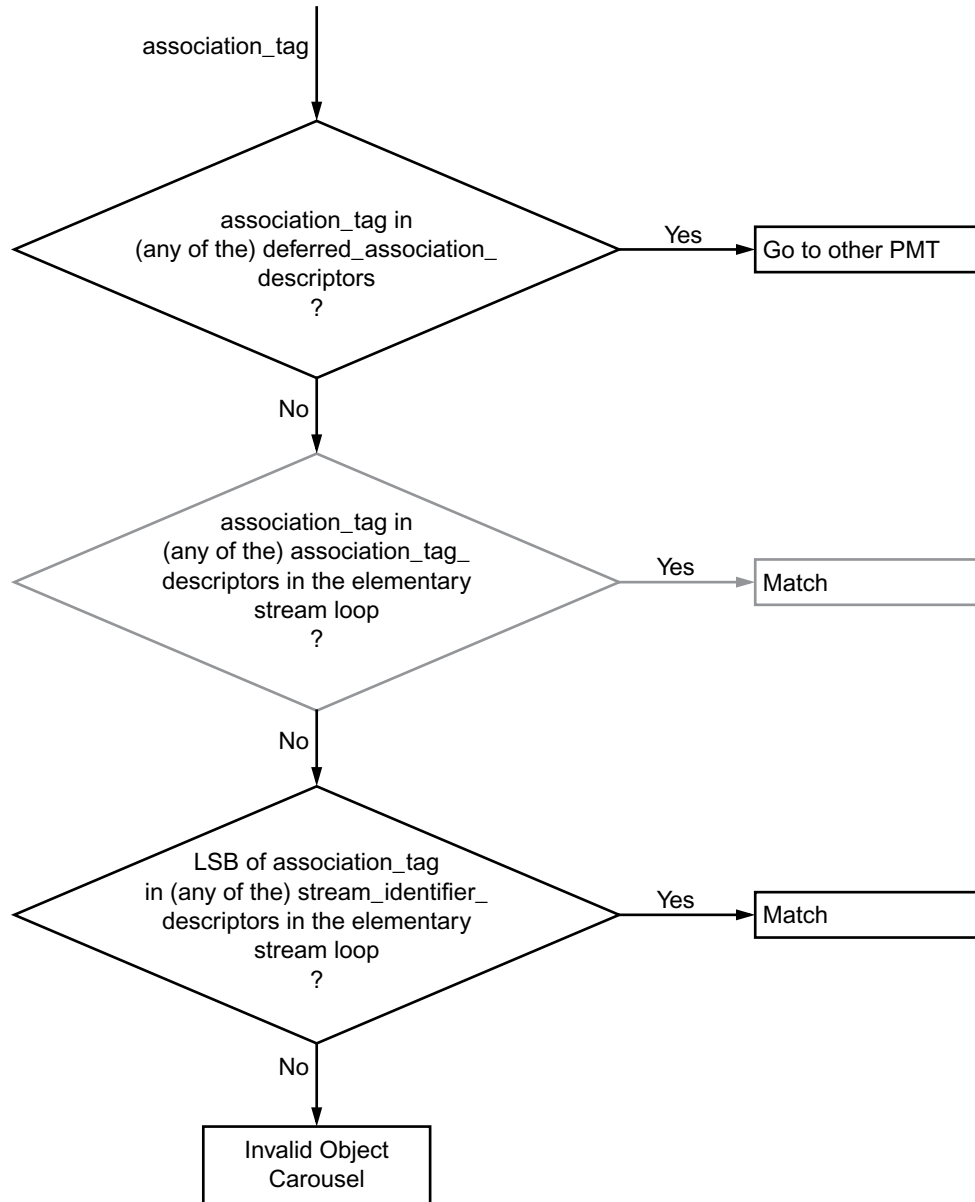


Figure B.1 : Object Carousel ES identification decision tree

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_tags_descriptor required by this specification is the adaptation of the ISO/IEC 13818-6 [26] descriptor defined in TR 101 202 [49]. This latter definition standardises a mechanism to signal the original network id.

When attempting to map an association_tag to an elementary stream the association_tag must first be checked against any deferred_association_tags_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_tags_descriptor then the matching process proceeds to the service indicated in that descriptor. The MHP terminal is not required to continue its search beyond this second service.

If the transport_stream_id field in the deferred_association_tags_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_tags_descriptor

The transport_stream_id field may take value 0x0000 in which case it shall be ignored in resolving the 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.

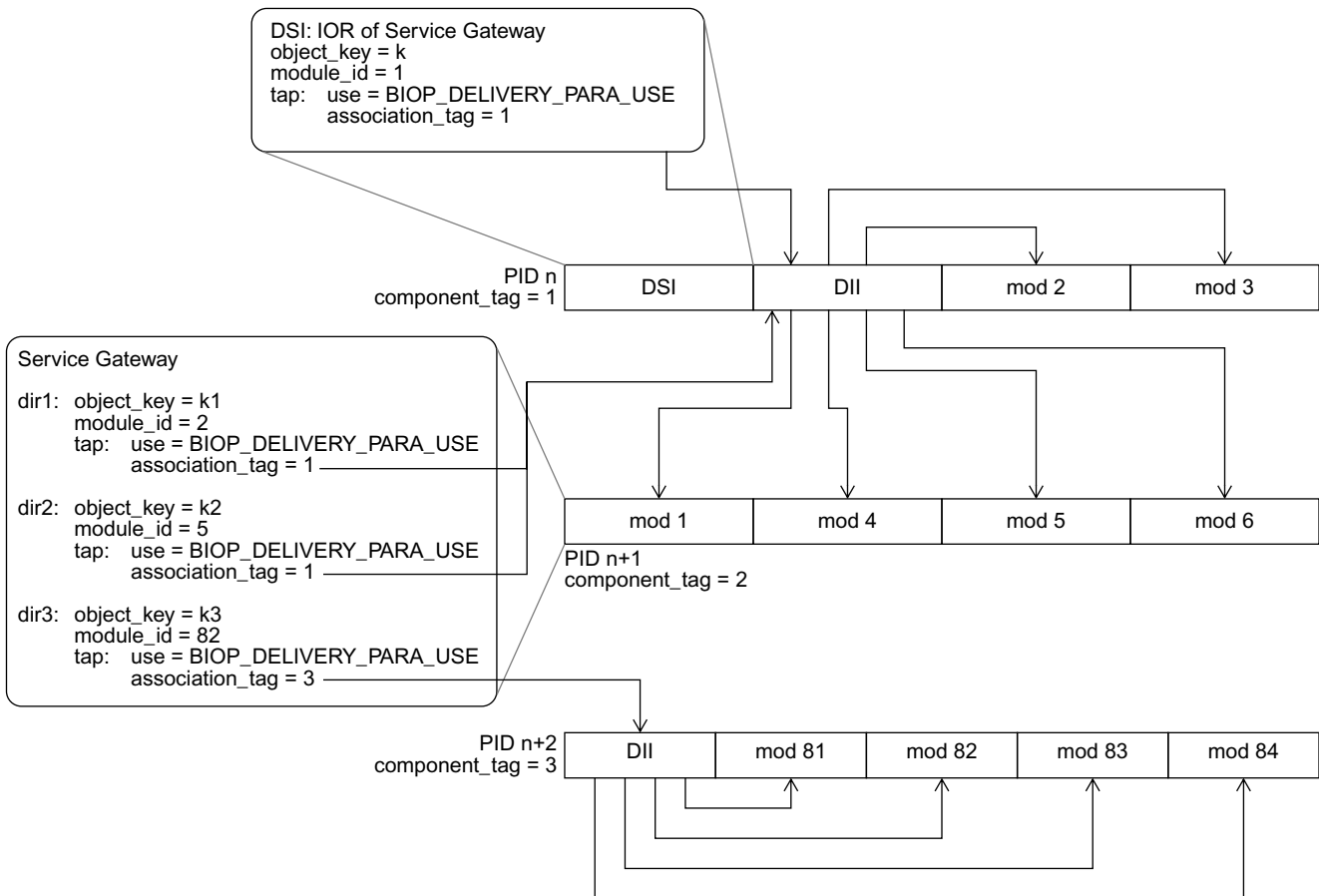


Figure B.2 : Example carousel

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. The circumstances under which this checking is required are 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 270). 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. http://java.sun.com/products/java-media/jmf/forDevelopers/playerguide/index.html .	
[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.	

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 object rendering as supported by DVB Text Layout Manager as described in U, "(normative): Extended graphics APIs" on page 653.

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 320.
- 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 329.
- For the content formats used to deliver fonts see 7.4, "Downloadable Fonts" on page 53.

D.2 Fonts

D.2.1 Embedded fonts

See G.4, "Resident fonts and text rendering" on page 329.

D.2.2 Downloaded fonts

D.2.2.1 Font technology

See 7.4, "Downloadable Fonts" on page 53.

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.

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.

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 `org.dvb.ui.FontFactory.createFont()`.

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 Low and high level rendering

Two levels of interface are addressed:

D.3.1.1 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.

`org.havi.ui.HDefaultTextLayoutManager` shall also be considered as supporting low level rendering except that implementations shall respect the rendering settings on the `HVisible` passed as argument to the render method (such as alignment, font and foreground colour).

D.3.1.2 High level rendering

More complex 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.3.2 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. The scope of this is much broader for "high level" rendering, where rules address text flow in addition to just string rendering.

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.3.2.1 High level rendering conceptual process

The conceptual rendering process can be described as follows:

- a) Based on:
 - the size of the object to render into
 - the characteristics of the font (e.g. see [D.3.3.1, "Font bounds" on page 306](#))
 - any automatic or application controlled insets (see [D.3.5, "Rendering within limits and insets" on page 308](#))
 calculate:
 - The maximum number of lines of text that may be rendered (see [D.3.8.1, "Number of lines" on page 311](#))
 - The width available for rendering on each line (see [D.3.9.1, "Available width" on page 314](#))
- b) Determine how the text to render flows, effectively defining a series of lines to render using:
 - The "logical" rules for calculating the width of rendered text (see [D.3.6, "'logical' text width rules" on page 309](#))
 - The available width for rendering (see [D.3.9.1, "Available width" on page 314](#))
 - The rules for breaking text (see [D.3.7, "Line breaking" on page 310](#))

- c) Determine where each line of text to render is placed vertically within the object (see D.3.8, "Positioning lines of text vertically within an object" on page 311).

This needs to consider that:

- If the number of lines of text to render (from step b) exceeds the maximum number of lines of text that may be rendered (from step a) "vertical truncation" may be required, i.e. discard some of the lines to render
- The positioning of lines to render is affected by the vertical justification setting of the object

- d) Determine the placement of individual characters in a line to render (see D.3.9, "Rendering lines of text horizontally" on page 314). This needs to consider that:

- Even having correctly applied the rules relating to the flow of text, in some extreme circumstances the length of the line of text to render can be wider than the available width for rendering (from step a). To handle this "horizontal truncation" may be required, i.e. discard some of the characters from the line to render
- The placement of characters in the line to render is affected by the horizontal justification setting of the object
- The placement of characters in the line to render should ensure sensible and consistent spacing between adjacent characters (see D.3.12, "Placing runs of characters & words" on page 316)

In addition special rules exist for handling tabulation (D.3.11, "Tabulation" on page 316) which need to be taken into account in the implementation of many of these steps.

Behaviour equivalent to the above is required to ensure conformant rendering. The order of these steps is illustrative and may vary between implementations for reasons of convenience or efficiency.

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.

D.3.3.1 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.

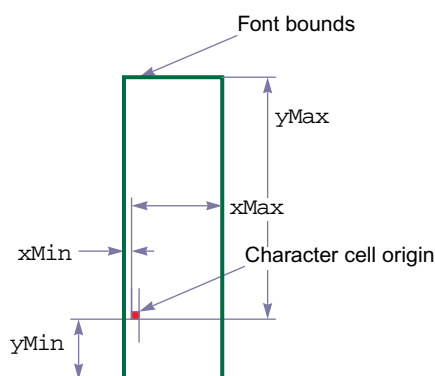


Figure D.1 : Font bounds

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.3.5.3, "Conversion of units" on page 308).

In "High level rendering" the text layout manager uses this information 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 and insets" on page 308.

D.3.3.2 "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.

The following table defines the pixel aspect ratios which shall be used when converting typographic pixels for rendering in the graphics system of an MHP terminal for each graphics device aspect ratio.

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 and insets

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.

D.3.5.1 Low level rendering

In "Low level rendering" the author is responsible for placing the text so that is not clipped.

D.3.5.2 High level rendering

In "High level rendering" text is automatically rendered with at least a sufficient inset from the object edge (derived from the font properties `xMin`, `yMin`, `xMax` and `yMax`) to ensure that all presented characters are completely rendered within the bounds of the object. Applications can further increase this minimum (font property determined) inset in the following ways:

- The `Insets` parameter on the `org.dvb.ui.DVBTextLayoutManager.render` method
- The `org.dvb.ui.DVBTextLayoutManager.setInsets` method

Each of these mechanisms can be used independently. Their effect is cumulative.

For simplicity, these application controlled insets are not described in the specification clauses that follow. In relation to the rendering process their behaviour is equivalent to a reduction in the size of the object within which the rendering processes acts.

D.3.5.3 Conversion of units

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 D.3.4, "Converting font metrics to display pixels" on page 307) this can be achieved by the following:

Vertically, for `yMin` and `yMax` (and for `top` and `bottom` insets):

$$Y_{\text{points}} = (\text{fontsize} \times Y_{\text{outlineResolution}}) / \text{outlineResolution}$$

$$Y_{\text{pixels}} = \text{roundupmag}(Y_{\text{points}})$$

Horizontally, for `xMin` and `xMax` (and for `left` and `right` insets):

$$X_{\text{points}} = (\text{fontsize} \times X_{\text{outlineResolution}}) / \text{outlineResolution}$$

$$X_{\text{pixels}} = \text{roundupmag}(X_{\text{points}} / \text{pixel_aspect_ratio})$$

outlineResolution : Is extracted from the font.

roundupmag: The function `roundupmag(A)` rounds the magnitude of A to the first integral number greater than or equal to A, whilst preserving the sign, e.g. `roundupmag(-4, 32) = -5`.

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 object. If justification is being used, only whitespace between words may be adjusted.

D.3.6.1.4 Letter spacing

Letter spacing allows for expansion/condensation of the character spacing for all of the characters in a object including whitespace.

NOTE: In printing, this is sometimes referred to as tracking.

D.3.6.2 Logical text width

The equation below shows how the width of a string of N characters is computed.

$$\text{logical width of N characters}_{\text{points}} = \text{div}((N-1) \times \text{letterspace}, 256) + \text{div}(\text{fontsize} \times \left(\sum_1^N \text{charSetWidth}[i] + \sum_1^{N-1} \text{kern}[i,i+1] \right), \text{metricsResolution})$$

$$\text{logical width of N characters}_{\text{pixels}} = \text{div}(\text{logical width of N characters}_{\text{points}} \times H, W)$$

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

D.3.7.1 Text wrapping setting is false

When the text wrapping setting is false, text shall break onto a new line only where a Carriage Return character is present in the text. The number of lines to render shall be equal to the number of Carriage Returns present, plus one.

NOTE: The "plus one" ensures that the last line of any text can be presented even if it has not been terminated with a Carriage Return.

D.3.7.2 Text wrapping setting is true

When the text wrapping setting is true, the text flow may break on to new lines at positions in addition to those caused by Carriage Return characters. These additional break positions are defined below. This behaviour is independent of the object's horizontal alignment settings and is considered to take place before any truncation (see D.3.8.2, "Truncation" on page 312).

The effect of text wrapping on the rendering process is equivalent to substituting "breaking characters" (defined in table D.4, "Special characters" on page 317) with Carriage Return characters at positions determined by the wrapping rules.

The behaviour of the wrapping process is described below:

- a) Based on the "logical" width of the text, receivers shall determine for each line, the first contiguous sequence of non-breaking characters that meets both of the following requirements:

- would not completely fit within the available width (see D.3.9.1, "Available width" on page 314)
- that follows one or more breaking characters

If such a sequence exists, the breaking character preceding it shall be replaced with a Carriage Return character.

- b) All trailing breaking characters shall be discarded from all lines of text. (Preceding breaking characters are not affected).

NOTE 1: In general use this identifies the word in a body of text that if rendered would cause the current line to spill outside of the text object. This word becomes the first word in the next line. However, if a line starts with a single word that is longer than the width of the object then the line is broken just before a following word.

Example 1: This illustrates the most common case where the word "won't" does not fit and so is broken to the next line.

Example 2: This illustrates the particular case of a long first word. In this case "is" is the first word that does not fit AND is preceded by a breaking character.

Example 2: This illustrates the consumption of trailing spaces. The line breaks just before the second "spaces" (so the next line starts with this word) and all trailing spaces (e.g. after the first "spaces,") are discarded).

NOTE 2: The importance of the removal of trailing breaking characters is particularly visible if the text is centred or right justified. For example, the text to be centred becomes "spaces" rather than " spaces".

After text wrapping the text truncation rules have to be considered. For example, in example 2 in figure D.3 the word "Antidisestablishmentarianism" will be truncated (see D.3.9.2, "Truncation" on page 314).

D.3.8 Positioning lines of text vertically within an object

D.3.8.1 Number of lines

Assuming that all characters in a line of text share a common baseline then, **regardless of the vertical alignment setting** the number of lines of text that can be presented within an object is:

$$\text{num_lines} = \text{floor}((\text{object_height} - (\text{yMin} + \text{yMax})) / \text{linespace}) + 1$$

All values are in pixels.

object_height: is the height of the component less any margin.

linespace: is an attribute of the object that defines the space between the baselines of consecutive lines of text. This is defined in units of points but is converted to pixels (see D.3.4, "Converting font metrics to display pixels" on page 307).

The function floor(A) rounds A to the first integral number less than or equal to A.

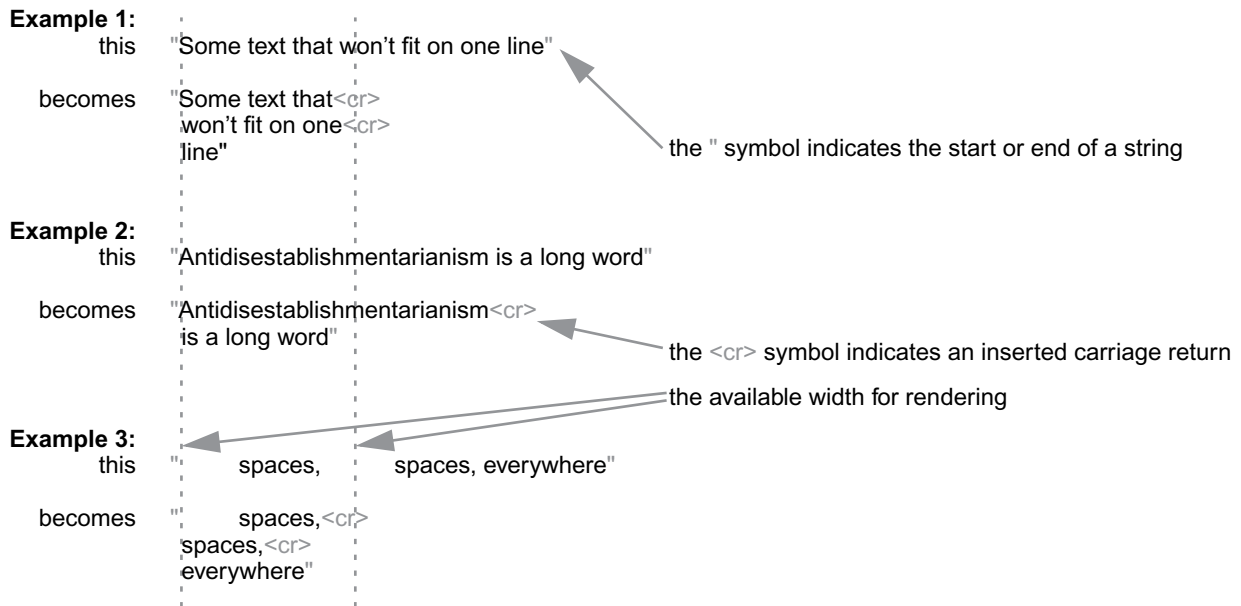


Figure D.3 : Text wrapping examples

D.3.8.2 Truncation

When the number of lines of text to be rendered exceeds the available height within the object then lines of text shall be discarded as follows:

- If the vertical alignment setting is VERTICAL_CENTER or VERTICAL_START_ALIGN then lines are discarded from the end of the text.
- If the vertical alignment setting is VERTICAL_END_ALIGN then lines are discarded from the beginning of the text.

This truncation will result in a number of lines that can be completely displayed within the object. These shall be presented according to D.3.8.3, "Positioning" on page 312.

See also D.3.10, "Text overflow" on page 315.

D.3.8.3 Positioning

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.

D.3.8.3.1 Vertical alignment setting is VERTICAL_START_ALIGN

When the text is top aligned the baseline of the top most line shall be at least y_{Max} inside the top of the object. The baselines of the subsequent lines of text shall be `linespace` apart.

D.3.8.3.2 Vertical alignment setting is VERTICAL_END_ALIGN

When the text is bottom aligned the baseline of the bottom most line shall be at least y_{Min} inside the bottom of the object. The baselines of the preceding lines of text shall be `linespace` apart.

D.3.8.3.3 Vertical alignment setting is VERTICAL_CENTER

When the text is vertically centred the positioning of the lines shall be such that the space above the first line of text equals the space below the last line of text (to allow for rounding the lower gap may be up to one pixel greater than the upper gap). The spacing between the baselines of text shall be `linespace`.

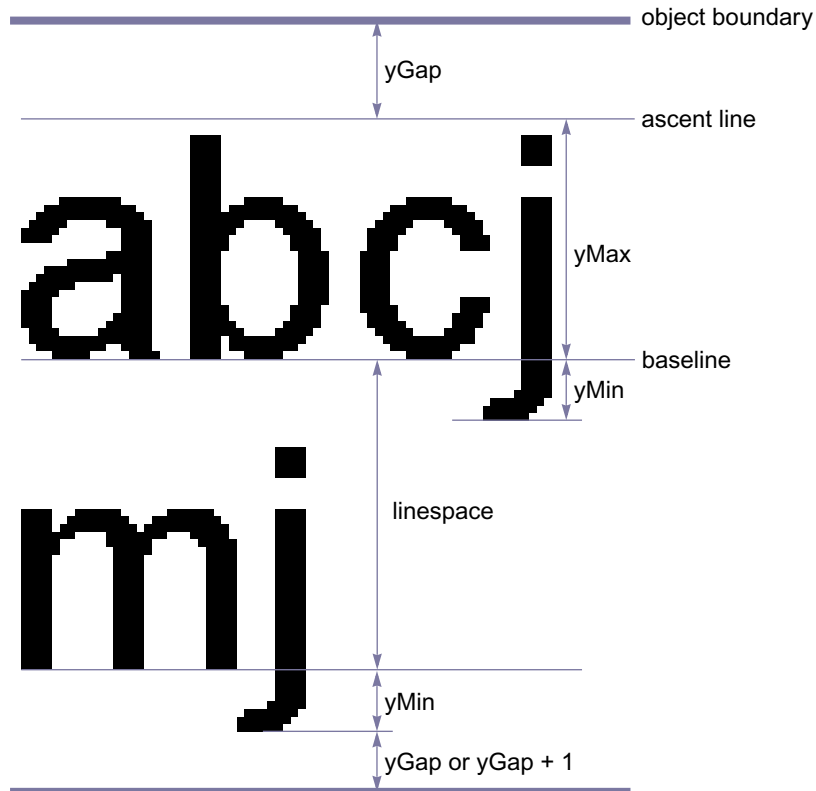


Figure D.4 : Vertical measures

D.3.8.3.4

Examples

vertical alignment->	VERTICAL_ START_ ALIGN	VERTICAL_ CENTER	VERTICAL_ END_ALIGN	String:
Right:	Three lines of	Three lines of	lines of text	Three<cr>lines of<cr>text
Wrong:	Three lines of text	Three lines of text	Three lines of text	
Wrong:		lines of		

Figure D.5 : Vertical positioning example 1

vertical alignment->	VERTICAL_ START_ ALIGN	VERTICAL_ CENTER	VERTICAL_ END_ALIGN	String:
Right:	Final CR	Final CR		Final CR<cr>
Wrong:		Final CR	Final CR	

Figure D.6 : Vertical positioning example 2

D.3.9 Rendering lines of text horizontally

D.3.9.1 Available width

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 the horizontal alignment setting** 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. This derivation of `available_width` shall be used with the "logical" text width rules to determine text flow.

D.3.9.2 Truncation

Where a line of text is too long to fit within the `available_width` of the object, the line shall be truncated.

NOTE: This addresses the case where text wrapping has not made the text fit. This can happen either because the text wrapping setting is false, or because a single word is too long to fit.

The result shall be as if the complete line were aligned appropriately on the object (taking into account the horizontal alignment setting) with only those characters which can be completely presented being rendered. That is those characters:

- whose origin is more than `xMin` right of the left hand edge of the object.
- whose right hand edge falls inside the right hand edge of the object.

For example:

- if the horizontal alignment setting is `HORIZONTAL_END_ALIGN` then a portion of text from the right end of the line will be rendered with its edge aligned to the right edge of the object.
- if the horizontal alignment setting is `HORIZONTAL_CENTER` the centre of the string will appear at the centre of the object with excess characters being truncated from each end.

See also [D.3.10, "Text overflow" on page 315](#).

D.3.9.3 Placement

- When the horizontal alignment setting is `HORIZONTAL_START_ALIGN` the origin of the left most character shall be `xMin` inside the left edge of the object.
- When the horizontal alignment setting is `HORIZONTAL_CENTER` the positioning of the characters shall be such that the gap to the left of the text equals the gap to the right (to within one pixel).
- When the horizontal alignment setting is `HORIZONTAL_END_ALIGN` the origin of the right most character shall be as necessary to ensure that it is completely visible when rendered.

NOTE 1: `xMin` allows for characters (such as capital "J") that extend to the left of their origin. `xMin` is a property of the font and so represents a consistent indent, i.e. it is independent of the first character.

NOTE 2: There are characters (e.g. fractional divisor) that can extend beyond the escapement indicated by their `charSetWidth` value. These can theoretically extend beyond the right hand edge of the object if the character is at the end of a word and that word is at the end of a line and the "logical width" of the line just fits the object. With the "fractional divisor" character this is a very unlikely combination of circumstances as this character is normally embedded within a "word". The handling of cases such as this by the MHP Terminal is implementation dependant.

horizontal alignment->	<code>HORIZONTAL_START_ALIGN</code>	<code>HORIZONTAL_CENTER</code>	<code>HORIZONTAL_END_ALIGN</code>	String:
Right:	This text is too lo	his text is too lon	is text is too long	This text is too long
Wrong:	This text is too lor	This text is too lor	his text is too long	
Wrong:		This text is too lo	This text is too lo	

Figure D.7 : Horizontal positioning examples

D.3.10 Text overflow

The `org.dvb.ui.TextOverflowListener.notifyTextOverflow` method is called if truncation occurs.

NOTE: This can be used by an application to alert the viewer or take other remedial action.

D.3.11 Tabulation

In left aligned text (the horizontal alignment setting is `HORIZONTAL_START_ALIGN`) tab stops are defined at intervals of `horizontalTabSpace` from the left edge of the object. By default the value of `horizontalTabSpace` is 56 points. "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).

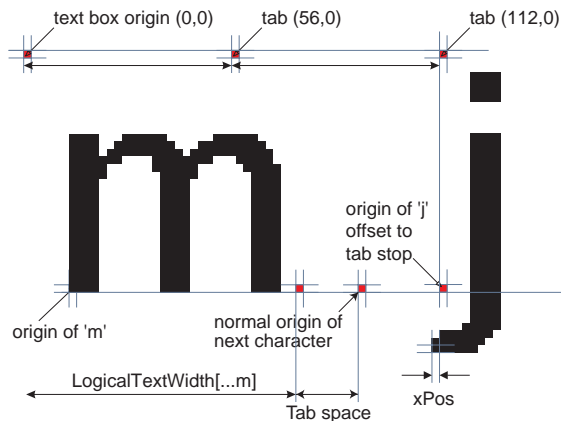


Figure D.8 : Effect of horizontal tabulation

- Tab characters only have meaning in left aligned text (the horizontal alignment setting is `HORIZONTAL_START_ALIGN`). 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 `abs(XMin)`. 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.3.12 Placing runs of characters & words

A run of characters starts from a well defined point:

- The start edge of the object
- A tab stop

After this origin the fine positioning of character cells and the gaps between words is not fully specified.

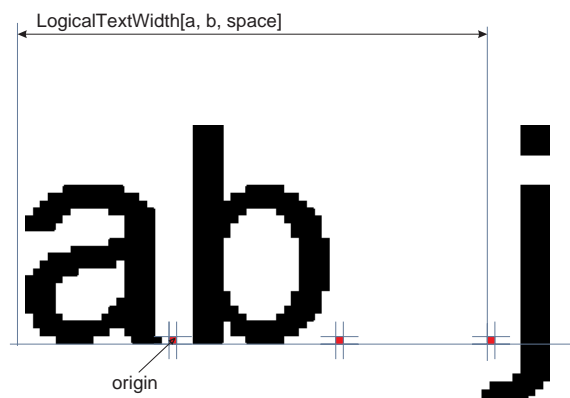


Figure D.9 : Calculation of character placement

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.

NOTE: The "logical" position of each character as determined by the "logical" rules is likely to be a non-integer. For renderers unable to support suitable sub-pixel positioning, e.g. a 1-bit/pixel bitmap renderer, this is impossible to implement. Thus, whilst the "logical" rules must be used to determine the flow of text, are not required to be observed for individual character placement within this flow and other strategies may be employed as long as the requirements above are satisfied.

D.3.13 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)	Character	Name	Breaking/ Non-breaking	Meaning
0x09	0x0009	Tab	Breaking	See "Tabulation" on page 316
0x0A	0x000A	Carriage Return	N/A	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 horizontal alignment setting.
0x0D	0x000D			
0x0D0A (note 1)	0x000D 0x000A			
0x20	0x0020	Space	Breaking	Spaces text by the width defined for the space character. When an object has the its text wrapping setting set to "true" lines may be broken at a space. See D.3.7, "Line breaking" on page 310 .
0xC2A0	0x00A0	Non-breaking Space	Non-breaking	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)
0xE28087	0x2007	Figure Space	Non-breaking	Can be used in a string of numerals as an alternative to using comma to denote "thousands". This character is not treated as a word boundary when deciding a position to break a line of text.
NOTE 1: The character sequence 0x0D0A shall be rendered identically to a single 0x0D.				

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 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 318)
- Format control mark-up (see D.4.4 on page 318)
- 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 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 : Codes defined for use in marked-up text files

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 version of this specification.			
NOTE 2: The required text encoding for this mark-up format is described in 11.2.11, "Text Encodings" on page 104.			

D.4.5 Future compatibility

Compatible extensions to the set of mark-up codes may be defined in future versions of this specification. For each the `markup_end_identifier` will be 32 (0x20) greater than the `markup_start_identifier`. MHP terminals 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	ISO 10646-1 [18] 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	ISO 10646-1 [18] 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	ISO 10646-1 [18] 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	ISO 10646-1 [18] 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

Table E.1 : Extended Latin set (Sheet 5 of 9)

Code	ISO 10646-1 [18] Character name
0x00EC	LATIN SMALL LETTER I WITH GRAVE
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

Table E.1 : Extended Latin set (Sheet 6 of 9)

Code	ISO 10646-1 [18] Character name
0x010F	LATIN SMALL LETTER D WITH CARON
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	ISO 10646-1 [18] 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	ISO 10646-1 [18] 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)

Code	ISO 10646-1 [18] Character name
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 >

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 DVB-J applications concurrently owning instances of `HScene` is not limited except by underlying platform resources like the total memory of the MHP terminal. The MHP terminal is required to support either the "Platforms Supporting a Restricted Multi-Window System" or the "Platforms Supporting a Full Multi-Window System" implementation scenarios as defined in `org.havi.ui.HSceneFactory`. The implementation scenario "Platforms Supporting a Single Window System" is not a valid choice for MHP.
- The MHP terminal is not required to provide a mechanism for the end-user to change user input focus between `HScenes`. Hence MHP applications wishing to receive user input focus must explicitly request it.
- 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 MHP terminal shall implement at least one `HScreen` which shall support at least one video, graphics and background device as defined immediately above.
- 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. These shall be reflected as `HScreenConfigurations` of the respective devices.

Optionally MHP terminals may also support square pixel `HGraphicsDevice` resolutions of 768x576 and 1024x576 for 4:3 and 16:9 displays respectively.

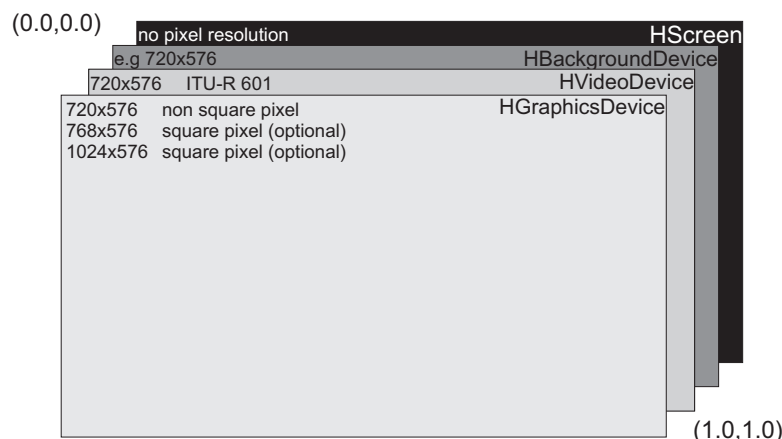


Figure G.1 : Required device resolutions

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 DVBBuffers objects shall support SRC and CLEAR and SRC_OVER. When SRC_OVER is used with DVBBuffers 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. DVBBuffers object with a type of TYPE_ADVANCED will produce a result as expected but those SRC_OVER operations are likely to be slow.
- DVBBuffers object created from a DVBBufferedImage with the type TYPE_ADVANCED shall perform SRC_OVER operations without approximations of the compositing rule.

G.1.3.1 Composition rules

MHP terminals are required to implement at least the SRC, SRC_OVER and CLEAR rules when compositing graphics over video and graphics over graphics.

G.1.4 Alpha capabilities

In the composition of the graphics (AWT/HAVi components) with background and video planes (see Figure 23, "Overview of AWT / HAVi plane composition" on page 194), 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% / 0x19-0xE6 it shall not be approximated as either 0% or 100% 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%.

Allowed approximations for the composition of graphics over other graphics are defined in 13.6.1.1.2, "Graphics over other graphics" on page 205.

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.

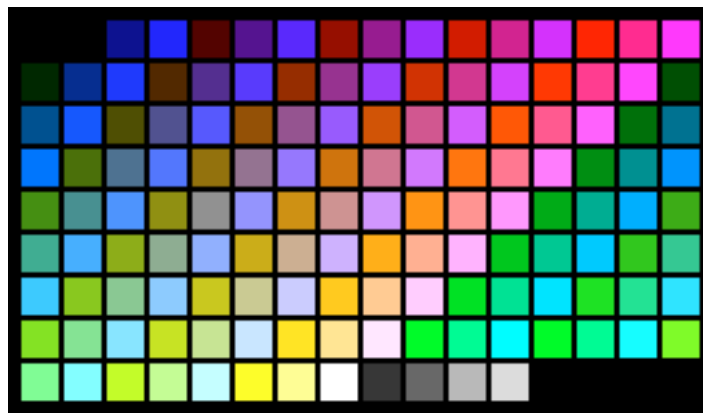


Figure G.2 : Opaque CLUT

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 9.4, "Inter application resource management" on page 74.

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 font *Tiresias* [81] 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

This table defines the minimum set of input events which shall be available to the set of running MHP applications if they are interested in receiving them. For DVB-J applications, this is described in more detail in 11.4.1.4, "Handling of input events" on page 111.

NOTE 1: They are not guaranteed to be always available to any one MHP application because another application running at the same time may have one of these events exclusively reserved. The application with focus (if any) always receives all of these events unless another application within the same service has requested and been granted exclusive access to one or more events. The process for event distribution for DVB-J applications is described in more detail in annex J, "(normative): DVB-J event API" on page 338.

NOTE 2: The user input device for an MHP terminal may support more events than this however this is implementation dependent. If more events than this are supported, it is equally implementation dependent whether the additional events are sent to MHP applications or sent to the MHP navigator. Events which are always sent to the MHP navigator may not be visible at all to MHP applications. For example, an MHP receiver using a conventional remote control will probably have program up/program down keys which are only ever sent to the navigator and cause service selection when received there.

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 218) 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]). 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 for TLS	Receivers shall support certificate key lengths up to and including 2048 bits for TLS (see 12.10, "Security on the return channel" on page 177).
Key lengths for Application Authentication (note 1)	Receivers shall support certificate key lengths up to and including 4096 bits for application authentication (see 12.2, "Authentication of applications" on page 145).
NOTE 1: It is not expected that key lengths as large as 4096 bits will be used initially.	

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, "Broadcast Channel Protocols" on page 45) and the MHP APIs defined to interface to these protocols and to control these interfaces.

Table G.5 : Minimum requirements for other resources for conformance purposes (Sheet 2 of 2)

Feature	Specification
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, "Interaction Channel Protocols" on page 48) 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	At least 4 kbytes. The content of this storage does not need to persist across interruptions in power.
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).
Applications in a service	MHP terminals shall not impose any arbitrary limit on the number of applications which they can support at the same time.
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): 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 constructors, methods or fields to classes and interfaces in the `org.dvb` namespace is **not** allowed.

NOTE: Restrictions on extensions to the `org.havi.ui` packages can be found in that specification.

Annex I (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

Declaration

```
public abstract 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)
    throws ClassNotFoundException
```

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 J (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.

J.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.

If an application has acquired exclusive access to receive an event through either of these mechanisms, it will not receive this same event through the other mechanism. An application can obtain exclusive access to an event through only one of the two mechanisms at a time.

The diagram below shows how the MHP terminal decides on the processing of an event (defined by a set of keychar/keycode, event id & modifiers). This includes deciding on the mechanisms by which the event will be dispatched and the appropriate Java class to encapsulate the event for each of these mechanisms.:

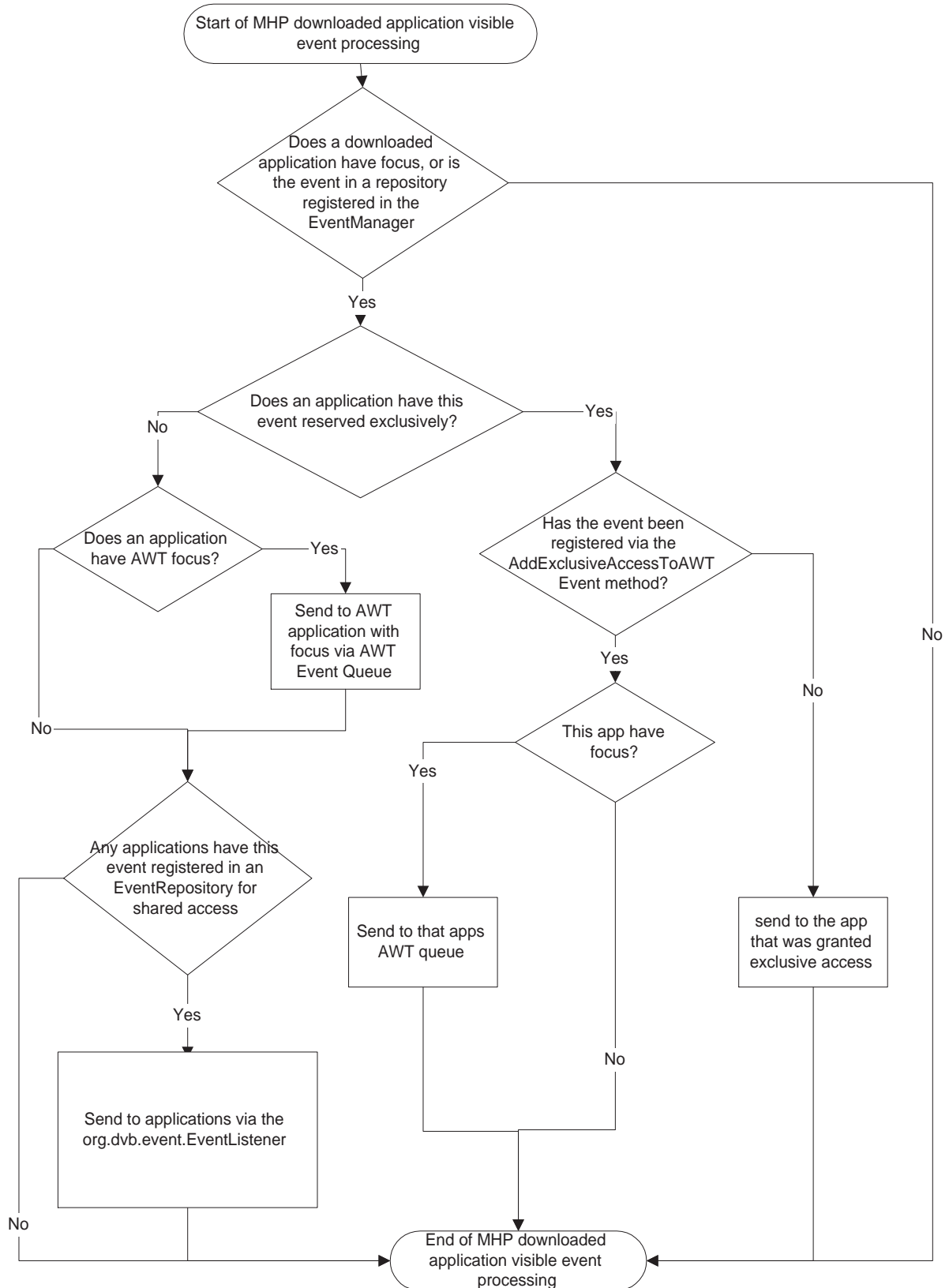


Figure 1 : The MHP downloaded application visible event distribution mechanism

J.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).

J.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.

J.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) {  
    ...  
  }  
}
```

J.4 Unicode

References to "Unicode" in the following API description shall be interpreted as references to [ISO 10646-1](#) [18].

J.5 Virtual keyboards

On platforms where key events are generated from a sequence of other (intermediate) key events, the intermediate key events shall not be visible to MHP applications by any mechanism. Examples of these intermediate key events include;

- For a virtual keyboard, the sequence of keys used to navigate around that keyboard (e.g. `VK_UP`, `VK_LEFT`, `VK_ENTER`)
- For multi-key press entry (as used in some mobile phones), the keys pressed before the final value is resolved.

Package org.dvb.event

Description

Provides access to user input events before they are processed through the event mechanism of the java.awt package.

The algorithm used for generating UserEvents by the MHP terminal when reporting user input to MHP applications shall be the same as that used for java.awt.event.KeyEvent. For example, pressing the Shift key will cause a KEY_PRESSED event with a VK_SHIFT keyCode, while pressing the 'a' key will result in a VK_A keyCode. After the 'a' key is released, a KEY_RELEASED event will be fired with VK_A, followed by a KEY_TYPED event with a keyChar value of 'A'.

Class Summary	
Interfaces	
UserEventListener	The listener interface for receiving user inputs.
Classes	
EventManager	The event manager allows an application to receive events coming from the user.
OverallRepository	This class defines a repository which initially contains all the user events which can be delivered to an application.
RepositoryDescriptor	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.
UserEvent	Represents a user event.
UserEventAvailableEvent	This event is sent to the resource status event listeners when user input events which had been exclusively reserved by an application are no longer exclusively reserved.
UserEventRepository	The application will use this class to define the events that it wants to receive.
UserEventUnavailableEvent	This event is sent to the resource status event listeners when user input events are exclusively reserved by an application.

org.dvb.event EventManager

Declaration

```
public 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.

Constructors

EventManager()

```
protected EventManager()
```

Constructor for instances of this class. This constructor is provided for the use of implementations and specifications which extend this specification. Applications shall not define sub-classes of this class. Implementations are not required to behave correctly if any such application defined sub-classes are used.

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.

The effect of multiple calls to this method by the same application with different instances of UserEventRepository shall be cumulative. If multiple calls to this method succeed in acquiring the

events in the specified repositories then the semantics of each successful method call shall be obeyed as specified.

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:

`addResourceStatusEventListener(ResourceStatusListener)` in interface `EventManager`

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.

The effect of multiple calls to this method by the same application with different instances of `UserEventRepository` shall be cumulative. If multiple calls to this method succeed in acquiring the events in the specified repositories then the semantics of each successful method call shall be obeyed as specified. Note that this can result in applications receiving the same event through more than one event listener.

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.

The effect of multiple calls to this method by the same application with different instances of UserEventRepository shall be cumulative. If multiple calls to this method succeed in acquiring the events in the specified repositories then the semantics of each successful method call shall be obeyed as specified. Note that this can result in applications receiving the same event through more than one event listener.

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:

`removeResourceStatusEventListener(ResourceStatusListener)` in interface `EventManager`

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

Declaration

```
public class OverallRepository extends UserEventRepository
```

```
java.lang.Object
|
+--org.dvb.event.RepositoryDescriptor
|
+--org.dvb.event.UserEventRepository
|
+--org.dvb.event.OverallRepository
```

All Implemented Interfaces:

org.davic.resources.ResourceProxy

Description

This class defines a repository which initially contains all the user events which can be delivered to an application. This includes all keycodes for which KEY_PRESSED and KEY_RELEASED events can be generated and all keychars for which KEY_TYPED events can be generated. Note that the set of keycodes and keychars which can be generated is dependent on the input devices of the MHP terminal. 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](#), [org.havi.ui.event.HKeyCapabilities](#)

Constructors

OverallRepository()

```
public OverallRepository()
```

The constructor for the repository. The name of the constructed instance (as returned by getName()) is implementation dependent.

OverallRepository(String)

```
public OverallRepository(java.lang.String name)
```

The constructor for the repository with a name.

Parameters:

`name` - the name to use for the repository

org.dvb.event RepositoryDescriptor

Declaration

```
public class RepositoryDescriptor implements org.davic.resources.ResourceProxy  
  
java.lang.Object  
|  
+--org.dvb.event.RepositoryDescriptor
```

All Implemented Interfaces:

org.davic.resources.ResourceProxy

Direct Known Subclasses:

UserEventRepository

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:

getClient() in interface RepositoryDescriptor

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, or about to be lost, user event belongs.

Returns:

String the name of the repository.

org.dvb.event

UserEvent

Declaration

```
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 the specification of 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, long)

```
public UserEvent(java.lang.Object source, int family, char keyChar, long when)
```

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 keys like the red key on a remote control, shall not generate `KEY_TYPED` events. `KEY_TYPED` events shall have no modifiers and hence shall not report any modifiers as being down.

Parameters:

`source` - the `EventManager` which is the source of the event

`family` - the event family.

`keyChar` - the character typed

`when` - a long integer that specifies the time the event occurred

Since:

MHP 1.0.1

UserEvent(Object, int, int, int, int, long)

```
public UserEvent(java.lang.Object source, int family, int type, int code, int modifiers,  
                 long when)
```

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.awt.event.KeyEvent` or `org.havi.ui.event.HRcEvent`.

`modifiers` - the modifiers active when the key was pressed. These have the same semantics as modifiers in `java.awt.event.KeyEvent`.

`when` - a long integer that specifies the time the event occurred

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

getWhen()

```
public long getWhen()
```

Returns the timestamp of when this event occurred.

Returns:
a long

Since:
MHP 1.0.2

getModifiers()

```
public int getModifiers()
```

Returns the modifiers flag for this event. This method shall return 0 for UserEvents constructed using a constructor which does not include an input parameter specifying the modifiers.

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. This method shall return false for UserEvents constructed using a constructor which does not include an input parameter specifying the modifiers.

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. This method shall return false for UserEvents constructed using a constructor which does not include an input parameter specifying the modifiers.

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. This method shall return false for UserEvents constructed using a constructor which does not include an input parameter specifying the modifiers.

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. This method shall return false for UserEvents constructed using a constructor which does not include an input parameter specifying the modifiers.

Returns:

whether the Shift modifier is down on this event

Since:

MHP 1.0.1

org.dvb.event UserEventAvailableEvent

Declaration

```
public class UserEventAvailableEvent extends org.davic.resources.ResourceStatusEvent
```

```
java.lang.Object
|
+-- java.util.EventObject
    |
    +-- org.davic.resources.ResourceStatusEvent
        |
        +-- org.dvb.event.UserEventAvailableEvent
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

This event is sent to the resource status event listeners when user input events which had been exclusively reserved by an application are no longer exclusively reserved. Where one change in user input event reservation results in instances of this event being sent to several applications, the following shall apply.

- Each application shall receive its own instance of the `UserEventRepository` object which forms the source to this event. Any changes made to that repository by any one application shall not impact the instance seen by any other application.
- Any application receiving an instance of this event is allowed to attempt to to exclusively reserve some of the newly available user events. In this situation, the normal resource management policy of the platform as described elsewhere in this specification shall be obeyed.
- Any applications which have registered for shared access to any of these user events shall start receiving those events following receipt of this event.

Since:

MHP 1.0.2

Constructors

UserEventAvailableEvent(Object)

```
public UserEventAvailableEvent(java.lang.Object source)
```

Constructor for the event.

Parameters:

`source` - a `UserEventRepository` which contains the events which stopped being exclusively reserved.

Since:

MHP 1.0.2

Methods

getSource()

```
public java.lang.Object getSource()
```

Returns a `UserEventRepository` which contains the events which were formerly exclusively reserved as passed into the constructor of the instance.

Overrides:

`org.davic.resources.ResourceStatusEvent.getSource()` in class
`org.davic.resources.ResourceStatusEvent`

Returns:

a `UserEventRepository`

Since:

MHP 1.0.2

org.dvb.event UserEventListener

Declaration

```
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

Declaration

```
public class UserEventRepository extends RepositoryDescriptor
```

```
java.lang.Object
|
+--org.dvb.event.RepositoryDescriptor
|
+--org.dvb.event.UserEventRepository
```

All Implemented Interfaces:

[org.davic.resources.ResourceProxy](#)

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 `KEY_PRESSED` type event without the `KEY_RELEASED` 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 reservation 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.

If an application loses exclusive access to a repository, it shall lose access to all events defined in that 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.

Unless stated otherwise, all constants used in the specification of this class are defined in `java.awt.event.KeyEvent` and its parent classes and not in this class.

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. After calling this method, the keycodes shall be present for both the KEY_PRESSED and KEY_RELEASED modes.

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. After calling this method, the keycodes shall be present for both the KEY_PRESSED and KEY_RELEASED modes.

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. After calling this method, the keycodes shall be present for both the KEY_PRESSED and KEY_RELEASED modes.

addKey(int)

```
public void addKey(int keycode)
```

Adds the specified keycode to the repository. Keycodes added in this way shall be listed in the list of user events returned by the `getUserEvent` method. If a key is already in the repository, this method has no effect. After calling this method, the keycode shall be present for both the KEY_PRESSED and KEY_RELEASED modes.

Parameters:

keycode - the key code.

addUserEvent(UserEvent)

```
public void addUserEvent(UserEvent event)
```

Adds the given user event to the repository. The values of the modifiers (if any) in the `UserEvent` shall be ignored by the MHP terminal. The value of the source used to construct the specified `UserEvent` shall be ignored by the MHP terminal when the `UserEventRepository` is used to specify events which an application wants to receive.

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. After calling this method, the keycodes shall not be present for both the KEY_PRESSED and KEY_RELEASED modes.

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. After calling this method, the keycodes shall not be present for both the KEY_PRESSED and KEY_RELEASED modes.

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. After calling this method, the keycodes shall not be present for both the KEY_PRESSED and KEY_RELEASED modes.

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. After calling this method, the keycode shall not be present for both the KEY_PRESSED and KEY_RELEASED modes.

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.

org.dvb.event

UserEventUnavailableEvent

Declaration

```
public class UserEventUnavailableEvent extends org.davic.resources.ResourceStatusEvent
```

```
java.lang.Object
|
+-- java.util.EventObject
|   |
|   +-- org.davic.resources.ResourceStatusEvent
|       |
|       +-- org.dvb.event.UserEventUnavailableEvent
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

This event is sent to the resource status event listeners when user input events are exclusively reserved by an application.

Each application shall receive its own instance of the `UserEventRepository` object which forms the source to this event. Any changes made to that repository by any one application shall not impact the instance seen by any other application.

Any applications which have registered for shared access to any of these user events shall stop receiving those user events following receipt of this event. If such user events become available again, a `UserEventAvailableEvent` shall be generated by the platform before any more of those user events are received by applications.

Since:

```
MHP 1.0.2
```

Constructors

UserEventUnavailableEvent(Object)

```
public UserEventUnavailableEvent(java.lang.Object source)
```

Constructor for the event.

Parameters:

`source` - a `UserEventRepository` which contains the events which were exclusively reserved.

Since:

```
MHP 1.0.2
```

Methods

getSource()

```
public java.lang.Object getSource()
```

Returns a `UserEventRepository` which contains the events which were exclusively reserved as passed into the constructor of the instance.

Overrides:

`org.davic.resources.ResourceStatusEvent.getSource()` in class
`org.davic.resources.ResourceStatusEvent`

Returns:

a `UserEventRepository`

Since:

MHP 1.0.2

Annex K (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

Declaration

```
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

Declaration

```
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)
    throws IOException
```

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)
    throws IOException
```

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 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	
Interfaces	
<code>UserPreferenceChangeListener</code>	An application wishing to be informed of any change to a user preference implements this interface.
Classes	
<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.
Exceptions	
<code>UnsupportedPreferenceException</code>	Thrown when a non-supported preference is used.

org.dvb.user Facility

Declaration

```
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

Declaration

```
public final class GeneralPreference extends Preference
```

```
java.lang.Object
|
+--org.dvb.user.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)
    throws IllegalArgumentException
```

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

Declaration

```
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

Declaration

```
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

Declaration

```
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

Declaration

```
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

Declaration

```
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.

When end-user preferences are read into a `Preference` object from the MHP terminal, the ordering of these values shall be as determined by the end-user, from most preferred to least preferred to the extent that this is known.

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. After this method returns, the values in the `Preference` object shall be the values of that user preference with any unsupported values from the `Facility` removed from that list. Note that the order of values returned here need not be the same as that returned by `read(Preference)`.

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)
    throws UnsupportedOperationException, IOException
```

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:

`UnsupportedOperationException` - 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

Declaration

```
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

M.1 Unicode

References to "Unicode" in the following API description shall be interpreted as references to [ISO 10646-1 \[18\]](#).

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	
Interfaces	
<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.

Class Summary	
<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.
Classes	
<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 <code>SIRequest.cancelRequest</code> 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.
Exceptions	
<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

Declaration

```
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)  
    throws IndexOutOfBoundsException
```

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. The value returned shall be the actual value used and is not limited to the values defined in `DescriptorTag`.

Returns:

The descriptor tag (the most common values are defined in the `DescriptorTag` interface)

See Also:

`DescriptorTag`

org.dvb.si DescriptorTag

Declaration

```
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

Declaration

```
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. The value returned shall be the actual value from the descriptor loop and is not limited to the set of values defined in `PMTStreamType`.

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

Declaration

```
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, SIRecoveryListener, short[])

```
public SIRequest retrievePMTElementaryStreams(short retrieveMode, java.lang.Object appData,
      SIRecoveryListener listener, short[] somePMTDescriptorTags)
      throws SIIllegalArgumentExcepTion
```

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: SIObjectNotInCacheEvent, SIObjectNotInTableEvent or SITableNotFoundEvent. This method will retrieve PMTElementaryStreams from the same sub-table version as this PMTService instance. If this version of the sub-table is no longer available, an SITableUpdatedEvent is returned.

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` - SIRecoveryListener 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, SIRecoveryListener, PMTElementaryStream

org.dvb.si PMTStreamType

Declaration

```
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

Declaration

```
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 SIBouquet
```

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. If the language returned by javax.tv.service.SIManager.getPreferredLanguage is one of those in the

multilingual_bouquet_name_descriptor, return the name in that language, otherwise return an implementation dependent selection between the names in the multilingual_bouquet_name_descriptor and the name in the 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. If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` is one of those in the multilingual_bouquet_name_descriptor, return the name in that language, otherwise return an implementation dependent selection between the names in the multilingual_bouquet_name_descriptor and the name in the bouquet_name_descriptor. If the descriptor is not present, "" is returned. If the string can be found but does not contain control codes for abbreviating it, the full string shall be 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)
    throws SIIllegalArgumentException
```

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 `SIBouquet`

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)
    throws SIIllegalArgumentException
```

This method defines extra semantics for the `SIInformation.retrieveDescriptors` method (second 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, short[]) in
interface SIBouquet
```

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)  
throws SIIllegalArgumentException
```

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. This method will retrieve `SIbouquetTransportStreams` from the same sub-table version as this `SIbouquet` instance. If this version of the sub-table is no longer available, an `SITableUpdatedEvent` is returned.

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

Declaration

```
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.

Constructors

SIDatabase()

```
protected SIDatabase()
```

This constructor is provided for the use of implementations and specifications which extend this specification. Applications shall not define sub-classes of this class. Implementations are not required to behave correctly if any such application defined sub-classes are used.

Methods

addBouquetMonitoringListener(SIMonitoringListener, int)

```
public void addBouquetMonitoringListener(SIMonitoringListener listener, int bouquetId)
    throws SIIllegalArgumentException
```

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)
    throws SIIllegalArgumentException
```

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 SI Database 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)
    throws SIIllegalArgumentException, SIInvalidPeriodException
```

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 SI Database 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)
    throws SIIllegalArgumentException
```

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 SI Database 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)
    throws SIIllegalArgumentException
```

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)
    throws SIIllegalArgumentException
```

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)
    throws SIIllegalArgumentException
```

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)
    throws SIIllegalArgumentException
```

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)
    throws SIIllegalArgumentException
```

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)
    throws SIIllegalArgumentException
```

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)
    throws SIIllegalArgumentException
```

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)
    throws SIIllegalArgumentException
```

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)
    throws SIIllegalArgumentException
```

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: SIObjectNotInCacheEvent or SITableNotFoundEvent

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)
    throws SIIllegalArgumentException
```

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: `SIOBJECTNOTINCACHEEVENT`, `SIOBJECTNOTINTABLEEVENT` or `SITABLENOTFOUNDEVENT`.

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 SRequest retrieveActualSITransportStream(short retrieveMode,
        java.lang.Object appData, SIRetrievalListener listener,
        short[] someDescriptorTags)
        throws SIIllegalArgumentException
```

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: SIOjectNotInCacheEvent SIOjectNotInTableEvent or SITableNotFoundEvent.

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 SRequest object

Throws:

SIIllegalArgumentException - thrown if the retrieveMode is invalid

See Also:

SRequest, SIRetrievalListener, SITransportStream, DescriptorTag

retrievePMTElementaryStreams(short, Object, SIRetrievalListener, DvbLocator, short[])

```
public SRequest retrievePMTElementaryStreams(short retrieveMode, java.lang.Object appData,
        SIRetrievalListener listener, org.davic.net.dvb.DvbLocator dvbLocator,
        short[] someDescriptorTags)
        throws SIIllegalArgumentException
```

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:

`SIIllegalArgumentExcpetion` - 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)
    throws SIIllegalArgumentExcpetion
```

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: `SIObjctNotInCacheEvent`, `SIObjctNotInTableEvent` or `SITableNotFoundEvent`.

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:

`SIIllegalArgumentExption` - 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)
    throws SIIllegalArgumentExption
```

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: `SIObjectNotInCacheEvent` `SIObjectNotInTableEvent` or `SITableNotFoundEvent`.

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:

`SIIllegalArgumentExption` - 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)
    throws SIIllegalArgumentException
```

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: `SIOBJECTNOTINCACHEEVENT`, `SIOBJECTNOTINTABLEEVENT` or `SITABLENOTFOUNDEVENT`.

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)
    throws SIIllegalArgumentException
```

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: `SIOBJECTNOTINCACHEEVENT`, `SIOBJECTNOTINTABLEEVENT` or `SITABLENOTFOUNDEVENT`.

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)
    throws SIIllegalArgumentException
```

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: `SIOBJECTNOTINCACHEEVENT`, `SITABLENOTFOUNDEVENT`.

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:

`SIIllegalArgumentExcpion` - 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)
    throws SIIllegalArgumentExcpion
```

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: `SIObjctNotInCacheEvent` `SIObjctNotInTableEvent` or `SITableNotFoundEvent`

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:

`SIIllegalArgumentExcpion` - 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 SRequest retrieveSIServices(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, int originalNetworkId, int transportStreamId,
    int serviceId, short[] someDescriptorTags)
    throws SIllegalArgumentException
```

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 `SIterator` 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: `SObjectNotInCacheEvent`, `SObjectNotInTableEvent` or `SITableNotFoundEvent`.

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 `SRequest` object

Throws:

`SIllegalArgumentException` - thrown if the `retrieveMode` is invalid or the numeric identifiers are out of range

See Also:

`SRequest`, `SIRetrievalListener`, `SIService`, `DescriptorTag`

retrieveSITimeFromTDT(short, Object, SIRetrievalListener)

```
public SRequest retrieveSITimeFromTDT(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener)
    throws SIllegalArgumentException
```

Retrieve information associated with time from the Time and Date Table (TDT) from the actual transport stream.

The `SIterator` 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: `SIObjctNotInCacheEvent` `SIObjctNotInTableEvent` or `SITableNotFoundEvent`.

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)
    throws SIIllegalArgumentException
```

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: `SIObjctNotInCacheEvent` `SIObjctNotInTableEvent` or `SITableNotFoundEvent`.

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 SRequest retrieveSITransportStreamDescription(short retrieveMode,
    java.lang.Object appData, SIRetrievalListener listener,
    short[] someDescriptorTags)
    throws SIIllegalArgumentException
```

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: SIObjctNotInCacheEvent, SIObjctNotInTableEvent or SITableNotFoundEvent.

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 SRequest object

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid

See Also:

`SRequest`, `SIRetrievalListener`, `SITransportStreamDescription`, `DescriptorTag`

org.dvb.si

SIEvent

Declaration

```
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`.

Where methods return values from a `short_event_descriptor`, the following algorithm shall be used where more than one such descriptor is present. If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` is one of those for which there is a `short_event_descriptor` then return the value from that descriptor. Otherwise return an implementation dependent selection between the values in the available `short_event_descriptors`.

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 seconds.

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. If the descriptor is not present, "" is returned. If the string can be found but does not contain control codes for abbreviating it, the full string shall be 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 SRequest retrieveSIService(short retrieveMode, java.lang.Object appData,  
    SIRetrievalListener listener, short[] someDescriptorTags)  
    throws SIllegalArgumentException
```

This method retrieves the SIService object representing the service the event, represented by this SEvent, is part of.

The SIterator 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: SObjectNotInCacheEvent SObjectNotInTableEvent or SITableNotFoundEvent.

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 SRequest object

Throws:

`SIllegalArgumentException` - thrown if the retrieveMode is invalid

See Also:

`SRequest`, `SIRetrievalListener`, `SIService`, `DescriptorTag`

org.dvb.si SIException

Declaration

```
public abstract class SIException extends java.lang.Exception
```

```
java.lang.Object  
|  
+-- java.lang.Throwable  
    |  
    +-- java.lang.Exception  
        |  
        +-- org.dvb.si.SIException
```

All Implemented Interfaces:

```
java.io.Serializable
```

Direct Known Subclasses:

```
SIIllegalArgumentException, SIInvalidPeriodException
```

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

Declaration

```
public class SIIllegalArgumentException extends SIIException
```

```
java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- org.dvb.si.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

Declaration

```
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)
    throws SIIllegalArgumentException
```

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)
    throws SIIllegalArgumentException
```

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 `someDescriptorTags` parameter are used for filtering the descriptors that are returned. Only those descriptors whose tag value is included in the `someDescriptorTags` array are retrieved, unless the `someDescriptorTags` 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

Declaration

```
public class SIInvalidPeriodException extends SIException
```

```
java.lang.Object  
|  
+-- java.lang.Throwable  
    |  
    +-- java.lang.Exception  
        |  
        +-- org.dvb.si.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

Declaration

```
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

Declaration

```
public class SILackOfResourcesEvent extends SIRetrievalEvent
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
|   |  
|   +-- org.dvb.si.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

Declaration

```
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 SIIInformation type returned with the getSIIInformationType 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 SIIInformation type returned with the getSIIInformationType 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 SIIInformation type returned with the getSIIInformationType method is NETWORK.

Returns:

the networkId or -2 if not applicable for this event

getOriginalNetworkID()

```
public int getOriginalNetworkID()
```

Returns the originalNetworkId of the SIIInformation objects This method is only applicable if the SIIInformation type returned with the getSIIInformationType 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 SIIInformation objects This method is only applicable if the SIIInformation type returned with the getSIIInformationType method is PMT_SERVICE, PRESENT_FOLLOWING_EVENT or SCHEDULED_EVENT.

Returns:

the serviceId or -2 if not applicable for this event

getSIIInformationType()

```
public byte getSIIInformationType()
```

Get the SIIInformation type of the information that has changed

Returns:

The SIIInformation 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

Declaration

```
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

Declaration

```
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

Declaration

```
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 SINetwork
```

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. If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` is one of those in the `multilingual_network_name_descriptor`, return the name in that language, otherwise return an implementation dependent selection between the names in the `multilingual_network_name_descriptor` and the name in the `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. If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` is one of those in the `multilingual_network_name_descriptor`, return the name in that language, otherwise return an implementation dependent selection between the names in the `multilingual_network_name_descriptor` and the name in the `network_name_descriptor`. If the descriptor is not present, "" is returned. If the string can be found but does not contain control codes for abbreviating it, the full string shall be 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)
    throws SIIllegalArgumentException
```

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
SINetwork
```

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)
    throws SIIllegalArgumentException
```

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, SIReivalListener, short[]) in
interface SINetwork
```

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.

`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, SIReivalListener, short[])`

retrieveSITransportStreams(short, Object, SIReivalListener, short[])

```
public SIRequest retrieveSITransportStreams(short retrieveMode, java.lang.Object appData,
    SIReivalListener listener, short[] someDescriptorTags)
    throws SIIllegalArgumentException
```

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. This method will retrieve SITransportStreams from the same sub-table version as this SINetwork instance. If this version of the sub-table is no longer available, an SITableUpdatedEvent is returned.

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:

`SIIllegalArgumentExcpetion` - thrown if the `retrieveMode` is invalid

See Also:

`SIRequest`, `SIRetrievalListener`, `SITransportStreamNIT`, `DescriptorTag`

org.dvb.si SINotInCacheEvent

Declaration

```
public class SINotInCacheEvent extends SIRetrievalEvent
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
|   |  
|   +-- org.dvb.si.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

Declaration

```
public class SIObjectNotInTableEvent extends SIRetrievalEvent
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
|   |  
|   +-- org.dvb.si.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

Declaration

```
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.

Constructors

SIRequest()

```
protected SIRequest()
```

This constructor is provided for the use of implementations and specifications which extend this specification. Applications shall not define sub-classes of this class. Implementations are not required to behave correctly if any such application defined sub-classes are used.

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

Declaration

```
public class SIRequestCancelledEvent extends SIRetrievalEvent
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
    |  
    +-- org.dvb.si.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

Declaration

```
public abstract class SIRetrievalEvent extends java.util.EventObject
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
|  
+-- org.dvb.si.SIRetrievalEvent
```

All Implemented Interfaces:

```
java.io.Serializable
```

Direct Known Subclasses:

```
SILackOfResourcesEvent, SINotInCacheEvent, SIOBJECTNotInTableEvent,  
SIRequestCancelledEvent, SISuccessfulRetrieveEvent, SITableNotFoundEvent,  
SITableUpdatedEvent
```

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

Declaration

```
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

Declaration

```
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

Declaration

```
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. If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` is one of those in the `multilingual_service_name_descriptor`, return the name in that language, otherwise return an implementation dependent selection between the names in the `multilingual_service_name_descriptor` and the name in the `service_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. If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` is one of those in the `multilingual_service_name_descriptor`, return the name in that language, otherwise return an implementation dependent selection between the names in the `multilingual_service_name_descriptor` and the name in the `service_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. If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` is one of those in the `multilingual_service_name_descriptor`, return the name in that language, otherwise return an implementation dependent selection between the names in the `multilingual_service_name_descriptor` and the name in the `service_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. If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` is one of those in the `multilingual_service_name_descriptor`, return the name in that language, otherwise return an implementation dependent selection between the names in the `multilingual_service_name_descriptor` and the name in the `service_descriptor`. If the descriptor is not present, "" is returned. If the string can be found but does not contain control codes for abbreviating it, the full string shall be 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 `SIService`

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 SRequest retrieveFollowingSIEvent(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, short[] someDescriptorTags)
    throws SIIllegalArgumentException
```

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: SIOjectNotInCacheEvent SIOjectNotInTableEvent or SITableNotFoundEvent.

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 SRequest object

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid

See Also:

`SRequest`, `SIRetrievalListener`, `SIEvent`, `DescriptorTag`

retrievePMTService(short, Object, SIRetrievalListener, short[])

```
public SRequest retrievePMTService(short retrieveMode, java.lang.Object appData,
    SIRetrievalListener listener, short[] someDescriptorTags)
    throws SIIllegalArgumentException
```

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: SIOjectNotInCacheEvent SIOjectNotInTableEvent or SITableNotFoundEvent.

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)
    throws SIIllegalArgumentException
```

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: `SIObjctNotInCacheEvent` `SIObjctNotInTableEvent` or `SITableNotFoundEvent`.

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)
    throws SIIllegalArgumentException, SIInvalidPeriodException
```

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

Declaration

```
public interface SIServiceType
```

Description

This interface defines constants corresponding to the different service types.

See Also:

```
getServiceType()
```

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

Declaration

public class `SISuccessfulRetrieveEvent` extends `SIRetrievalEvent`

```

java.lang.Object
|
+-- java.util.EventObject
    |
    +-- org.dvb.si.SIRetrievalEvent
        |
        +-- 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:

`SIRetrievalListener`

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

Declaration

```
public class SITableNotFoundEvent extends SIRetrievalEvent
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
    |  
    +-- org.dvb.si.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

Declaration

```
public class SITableUpdatedEvent extends SIRetrievalEvent
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
    |  
    +-- org.dvb.si.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

Declaration

```
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

Declaration

```
public interface SITransportStream extends SIInformation
```

All Superinterfaces:

```
SIInformation
```

All Known Subinterfaces:

```
SITransportStreamBAT, SITransportStreamNIT
```

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 SRequest retrieveSIServices(short retrieveMode, java.lang.Object appData,  
    SIRetrievalListener listener, short[] someDescriptorTags)  
    throws SIllegalArgumentException
```

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 SIterator 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 SRequest object

Throws:

`SIllegalArgumentException` - thrown if the retrieveMode is invalid

See Also:

`SRequest`, `SIRetrievalListener`, `SIService`, `DescriptorTag`

org.dvb.si

SITransportStreamBAT

Declaration

```
public interface SITransportStreamBAT extends SITransportStream
```

All Superinterfaces:

```
SIIInformation, 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

Declaration

```
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

Declaration

```
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

Declaration

```
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)
    throws SIIllegalArgumentException
```

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:

`SIIllegalArgument`Exception - thrown if `offset` and/or `offset+length-1` is not a valid index in `dvbSIText`.

org.dvb.si

TextualServiceIdentifierQuery

Declaration

```
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	
Interfaces	
<code>BackgroundVideoPresentationControl</code>	A control to support the setting and querying of the video presentation for background players.
<code>DVBMediaSelectControl</code>	<code>DVBMediaSelectControl</code> extends <code>MediaSelectControl</code> allowing the selection of different kinds of content in a running <code>Player</code> .
<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.
Classes	
<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>CAStopEvent</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.

Class Summary

<code>SubtitleNotSelectedEvent</code>	Report that subtitles are not now selected.
<code>SubtitleSelectedEvent</code>	Report that subtitles are now selected.
<code>VideoFormatEvent</code>	The base class for all other events relating to changes in video format
<code>VideoTransformation</code>	VideoTransformation objects express video transformations, i.e.
Exceptions	
<code>CAException</code>	This exception is thrown when access to a media stream is denied by the CA system.

org.dvb.media

ActiveFormatDescriptionChangedEvent

Declaration

```
public class ActiveFormatDescriptionChangedEvent extends VideoFormatEvent
```

```
java.lang.Object
|
+-- java.util.EventObject
    |
    +-- org.dvb.media.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 `@link VideoFormatControl` and shall be the value passed into the constructor of the event.

org.dvb.media AspectRatioChangedEvent

Declaration

```
public class AspectRatioChangedEvent extends VideoFormatEvent
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
    |  
    +-- org.dvb.media.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 and shall be the value passed into the constructor of the event.

org.dvb.media

BackgroundVideoPresentationControl

Declaration

```
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

Declaration

```
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

Declaration

```
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, int, int, int, MediaLocator)

```
public CAStopEvent(javax.media.Controller source, int previous, int current, int target,
    javax.media.MediaLocator stream)
```

Construct an event.

Parameters:

`source` - the controller which was presenting the service

`stream` - the locator of the stream from which access has been withdrawn.

`previous` - the previous state of the controller

`current` - the current state of the controller

`target` - the target state of the controller

Methods

getStream()

```
public javax.media.MediaLocator getStream()
```

This method returns the stream from which access has been withdrawn.

Returns:

the locator for the stream concerned

org.dvb.media DFCChangedEvent

Declaration

```
public class DFCChangedEvent extends VideoFormatEvent
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
|   |  
|   +-- org.dvb.media.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 and shall be the value passed into the constructor of the event.

org.dvb.media DripFeedDataSource

Declaration

```
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 a locator of the following text representation: "dripfeed://".

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 the drip feed mode.

Methods

connect()

```
public void connect()
    throws IOException
```

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.

start()

```
public void start()
    throws IOException
```

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()
    throws IOException
```

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

Declaration

```
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.

Methods

implies(Permission)

```
public boolean implies(java.security.Permission p)
```

Checks if the specified permission is "implied" by this object.

Since name and actions aren't used, the only check needed is whether p is also a DripFeedPermission.

Overrides:

```
java.security.BasicPermission.implies(java.security.Permission) in class  
java.security.BasicPermission
```

Parameters:

p - the permission to check against.

Returns:

true if the passed permission is equal to or implied by this permission, false otherwise.

org.dvb.media

DVBMediaSelectControl

Declaration

```
public interface DVBMediaSelectControl extends javax.tv.media.MediaSelectControl
```

All Superinterfaces:

```
javax.media.Control, javax.tv.media.MediaSelectControl
```

Description

DVBMediaSelectControl extends MediaSelectControl allowing the selection of different kinds of content in a running Player. The extension is to allow the selection in a single operation of all the media service components in a service without needing knowledge about which media service components are present in that service.

Since:

MHP 1.0.2

See Also:

```
javax.tv.media.MediaSelectControl
```

Methods

selectServiceMediaComponents(Locator)

```
public void selectServiceMediaComponents (javax.tv.locator.Locator l)
    throws InvalidLocatorException, InvalidServiceComponentException, Insufficient
    ResourcesException
```

Selects for presentation the media service components from a service. If some content is currently playing, it is replaced in its entirety by the media service components from the specified service. This is an asynchronous operation that is completed upon receipt of a MediaSelectEvent. Note that for most selections that imply a different time base or otherwise change synchronization relationships, a RestartingEvent will be posted by the Player. The rules for deciding which media service components shall be presented are defined in the main body of this specification.

Parameters:

l - the locator for a service

Throws:

InvalidLocatorException - If the locator provided does not reference a service.

InvalidServiceComponentException - If the locator provided does not reference a service which contains at least one media service component

InsufficientResourcesException - If the operation cannot be completed due to a lack of system resources.

org.dvb.media

NoComponentSelectedEvent

Declaration

```
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, int, int, int, MediaLocator)

```
public NoComponentSelectedEvent(javax.media.Controller source, int previous, int current,
    int target, 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
- `previous` - the previous state of the controller
- `current` - the current state of the controller
- `target` - the target state of the controller

Methods

getStream()

```
public javax.media.MediaLocator getStream()
```

This method returns the stream whose presentation has stopped

Returns:

the locator for the stream concerned

org.dvb.media PresentationChangedEvent

Declaration

```
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 locator for the stream now being presented

org.dvb.media ServiceRemovedEvent

Declaration

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, int, int, int, MediaLocator)

```
public ServiceRemovedEvent(javax.media.Controller source, int previous, int current,
    int target, 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

`previous` - the previous state of the controller

`current` - the current state of the controller

`target` - the target state of the controller

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

Declaration

```
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, int, int, int, MediaLocator)

```
public StopByResourceLossEvent(javax.media.Controller source, int previous, int current,
    int target, 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
- `previous` - the previous state of the controller
- `current` - the current state of the controller
- `target` - the target state of the controller

Methods

getStream()

```
public javax.media.MediaLocator getStream()
```

This method returns the stream which was being presented

Returns:

the locator for the stream concerned

org.dvb.media SubtitleAvailableEvent

Declaration

```
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. The platform shall always pass in the JMF player presenting the subtitles.

Methods

getSource()

```
public java.lang.Object getSource()
```

Return the JMF player which is the source of the event.

Overrides:

java.util.EventObject.getSource() in class java.util.EventObject

Returns:

the source of the event. This shall be the JMF player passed in to the constructor of the event.

org.dvb.media SubtitleListener

Declaration

```
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

Declaration

```
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. The platform shall always pass in the JMF player presenting the subtitles.

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. This shall be the JMF player passed in to the constructor of the event.

org.dvb.media SubtitleNotSelectedEvent

Declaration

```
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. The platform shall always pass in the JMF player presenting the subtitles.

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. This shall be the JMF player passed in to the constructor of the event.

org.dvb.media SubtitleSelectedEvent

Declaration

```
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. The platform shall always pass in the JMF player presenting the subtitles.

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. This shall be the JMF player passed in to the constructor of the event.

org.dvb.media SubtitlingEventControl

Declaration

```
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

Declaration

```
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.

isPlatform()

```
public boolean isPlatform()
```

Test if control over the decoder format conversions is being managed by the platform as defined by DFC_PLATFORM.

Returns:

true if control over the decoder format conversions is being managed by the platform, false otherwise

See Also:

[DFC_PLATFORM](#)

removeVideoFormatListener(VideoFormatListener)

```
public void removeVideoFormatListener(VideoFormatListener l)
```

Remove a listener for VideoFormatChangedEvents

Parameters:

l - the listener to remove

org.dvb.media VideoFormatEvent

Declaration

```
public abstract class VideoFormatEvent extends java.util.EventObject
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
|  
+-- org.dvb.media.VideoFormatEvent
```

All Implemented Interfaces:

java.io.Serializable

Direct Known Subclasses:

[ActiveFormatDescriptionChangedEvent](#), [AspectRatioChangedEvent](#),
[DFCChangedEvent](#)

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. The platform shall always pass in the JMF Player presenting the video whose format changed.

org.dvb.media

VideoFormatListener

Declaration

```
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

Declaration

```
public interface VideoPresentationControl extends javax.media.Control
```

All Superinterfaces:

`javax.media.Control`

All Known Subinterfaces:

`BackgroundVideoPresentationControl`

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. Note that when the MHP terminal is in DFC_PROCESSING_PAN_SCAN, the return value of this method will be out of date almost as soon as the method has returned.

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 results are implementation dependent. 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 clipping.

org.dvb.media VideoTransformation

Declaration

```
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 or if clipping is disabled.

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 380) 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 (see 4.6.1 "Use of control codes in names" in ETR 211 [11]).

O.2.1.2 `getServiceType`

Returns the ServiceType according to the mapping defined in section O.2.3, "`javax.tv.service.ServiceType`" on page 511.

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`

If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` corresponds to the language of a multilingual component descriptor or data broadcast descriptor in the EIT, return the description in that language. Otherwise return an implementation dependent selection between the descriptions available in the EIT.

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, multilingual component descriptor or 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 511.

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.RatingDimension" on page 513.

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 513.

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 512

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 513. These objects shall not implement the ServiceNumber interface.

O.2.8.1 getLongName

If the language returned by javax.tv.service.SIManager.getPreferredLanguage corresponds to the language of a multilingual service descriptor in the SDT, return the service name in that language. Otherwise return an implementation dependent selection between the descriptors available in the SDT.

O.2.8.2 getServiceType

Returns the ServiceType according to the mapping defined in section O.2.3, "javax.tv.service.ServiceType" on page 511.

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

If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` corresponds to the language of a multilingual component descriptors or data broadcast descriptors in the EIT present/following table then the information for the ServiceComponents shall be retrieved from those descriptors. Otherwise the information for the ServiceComponents shall be returned from an implementation dependent selection between the descriptors available 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.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

If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` corresponds to the language of a multilingual service descriptor or service descriptor in the SDT, return the provider name from that descriptor. Otherwise return an implementation dependent selection from the descriptors available in the SDT.

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 513.

O.2.13.1 `getBouquetID`

Returns the integer DVB SI Bouquet ID value.

O.2.13.2 `getName`

If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` corresponds to the language of a multilingual bouquet name descriptor in the BAT, return the name in that language. Otherwise return an implementation dependent selection from the descriptors available in the BAT.

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`

If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` corresponds to the language of a multilingual network name descriptor in the NIT, return the name in that language. Otherwise return an implementation dependent selection from the descriptors available in the NIT.

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 513.

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`

If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` corresponds to the language of a short event descriptor in the EIT, return the name from that descriptor. Otherwise return an implementation dependent selection from the descriptors available in the EIT.

O.2.16.5 `retrieveDescription`

If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` corresponds to the language of a short event descriptor in the EIT, return the description from that descriptor. Otherwise return an implementation dependent selection from the descriptors available in the EIT.

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 515.

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 3 (i.e. the value in the descriptor + 3). In other cases when the parameter is "DVB Age based rating" returns an empty string.

NOTE: Applications wishing to present this information in languages other than English should use the `getRatingLevel()` method and perform their own encoding of this for end-user presentation.

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 3 (i.e. the value in the descriptor + 3), as its substring.

NOTE: Applications wishing to present this information in languages other than English should use the `getRatingLevel()` method and perform their own encoding of this for end-user presentation.

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, i.e. by using the 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	
Interfaces	
<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>NPTStatusEvent</code> and <code>NPTRateChangedEvent</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.
Classes	
<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 to listeners on a <code>DSMCCStream</code> object when NPT newly appears for that DSMCC stream when 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 to listeners on a <code>DSMCCStream</code> object when NPT which was present for that DSMCC stream is removed.

Class Summary	
<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.
Exceptions	
<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>InsufficientResourcesException</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

Declaration

```
public abstract class AsynchronousLoadingEvent extends java.util.EventObject
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
|  
+-- org.dvb.dsmcc.AsynchronousLoadingEvent
```

All Implemented Interfaces:

java.io.Serializable

Direct Known Subclasses:

InvalidFormatEvent, InvalidPathnameEvent, LoadingAbortedEvent, MPEGDeliveryErrorEvent, NotEntitledEvent, ServerDeliveryErrorEvent, ServiceXFRErrorEvent, SuccessEvent

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

Declaration

```
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

Declaration

```
public class DSMCCEXception extends java.io.IOException
```

```
java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- java.io.IOException
            |
            +-- org.dvb.dsmcc.DSMCCEXception
```

All Implemented Interfaces:

```
java.io.Serializable
```

Direct Known Subclasses:

```
IllegalObjectTypeException, InsufficientResourcesException,
InvalidAddressException, InvalidFormatException, InvalidPathNameException,
MPEGDeliveryException, NotEntitledException, NothingToAbortException,
NotLoadedException, ServerDeliveryException, ServiceXFRException,
UnknownEventException
```

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

Declaration

```
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 file pathname.

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 file pathname.

Methods

abort()

```
public void abort()  
    throws NothingToAbortException
```

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)
    throws InsufficientResourcesException
```

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.

NOTE: The algorithm used for this change monitoring is implementation dependent. In some implementations, this exception will always be thrown. In other implementations, it will never be thrown. In other implementations, whether it is thrown or not will depend on the complexity and design of the object carousel in which the object is carried. Even where no exception is thrown, implementations are not required to detect all possible forms in which an object may change.

Parameters:

`listener` - the `ObjectChangeListener` to be notified .

Throws:

`InsufficientResourcesException` - if there are not sufficient resources to monitor the object for changes.

asynchronousLoad(AsynchronousLoadingEventListener)

```
public void asynchronousLoad(AsynchronousLoadingEventListener l)
    throws InvalidPathNameException
```

This method is used to asynchronously load a carousel object. This method can fail either asynchronously with an event or synchronously with an exception. When it fails synchronously with an exception, no event shall be sent to the listener. 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`, `LoadingAbortedEvent`

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 where the chain and the corresponding signature file have been validated by the MHP terminal. Valid chains do not need to originate from root certificates known to the MHP terminal, e.g. self signing of data files. Applications should note that calls to this method may take some time. 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

isLoaded()

```
public boolean isLoaded()
```

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)
    throws InvalidPathNameException
```

Asynchronous loading of the directory entry information. Calling this is equivalent of calling the method `asynchronousLoad` on the parent directory of a `DSMCCObject`. This method can fail either asynchronously with an event or synchronously with an exception. When it fails synchronously with an exception, no event shall be sent to the listener.

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()
```

```
throws InvalidFormatException, InterruptedException, MPEGDeliveryException,
ServerDeliveryException, InvalidPathNameException, NotEntitledException, ServiceXFRException
```

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 `InterruptedException` 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:

`InterruptedException` - 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()
```

```
throws NotLoadedException
```

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

Declaration

```
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](#) The BIOP::Stream message shall be read from the network once only, before the constructor of this class returns. Hence methods which return information from that message shall not be affected by any subsequent changes to that information.

Constructors

DSMCCStream(DSMCCObject)

```
public DSMCCStream(DSMCCObject aDSMCCObject)
    throws NotLoadedException, IllegalObjectTypeException
```

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)
    throws IOException, IllegalObjectTypeException
```

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.

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)
    throws IOException, IllegalObjectTypeException
```

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.

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()
    throws MPEGDeliveryException
```

This function is used to get the current NPT in milliseconds. Implementations are not required to continuously monitor for NPT. In implementations which do not continuously monitor, this method will block while the current NPT is retrieved from the stream.

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()  
    throws MPEGDeliveryException
```

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

Throws:

throws - `MPEGDeliveryException` if there's an error in retrieving NPT reference descriptors

`MPEGDeliveryException`

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

Declaration

```
public class DSMCCStreamEvent extends DSMCCStream
```

```
java.lang.Object
|
|--org.dvb.dsmcc.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. The `BIOP::StreamEvent` message shall be read from the network once only, before the constructor of this class returns. Hence methods which return information from that message shall not be effected by any subsequent changes to that information.

The subscribe method only verifies that the event name can be bound to an `eventId` but it does not require that the stream event descriptors for that event id can be received at that moment. While the event listener is registered the MHP terminal shall filter the stream event descriptors as specified in Monitoring stream eventt in the main body of the specification.

Constructors

DSMCCStreamEvent(DSMCCObject)

```
public DSMCCStreamEvent(DSMCCObject aDSMCCObject)
    throws NotLoadedException, IllegalObjectTypeException
```

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)
    throws IOException, IllegalObjectTypeException
```

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 DSMCC StreamEvent.

DSMCCStreamEvent(String, String)

```
public DSMCCStreamEvent(java.lang.String path, java.lang.String name)
    throws IOException, IllegalObjectTypeException
```

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)
    throws UnknownEventException, InsufficientResourcesException
```

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)
    throws UnknownEventException
```

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)
    throws UnknownEventException
```

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

Declaration

```
public class IllegalObjectTypeException extends DSMCCException
```

```
java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- java.io.IOException
            |
            +-- org.dvb.dsmcc.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

Declaration

```
public class InsufficientResourcesException extends DSMCCEException
```

```
java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- java.io.IOException
            |
            +-- org.dvb.dsmcc.DSMCCEException
                |
                +-- 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

Declaration

```
public class InvalidAddressException extends DSMCCEException
```

```
java.lang.Object  
|  
+--java.lang.Throwable  
|  
+--java.lang.Exception  
|  
+--java.io.IOException  
|  
+--org.dvb.dsmcc.DSMCCEException  
|  
+--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

Declaration

```
public class InvalidFormatEvent extends AsynchronousLoadingEvent
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
|   |  
   +-- org.dvb.dsmcc.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

Declaration

public class **InvalidFormatException** extends `DSMCCEException`

```
java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- java.io.IOException
            |
            +-- org.dvb.dsmcc.DSMCCEException
                |
                +-- 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

Declaration

```
public class InvalidPathnameEvent extends AsynchronousLoadingEvent
```

```
java.lang.Object
|
+-- java.util.EventObject
    |
    +-- org.dvb.dsmcc.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

Declaration

```
public class InvalidPathNameException extends DSMCCException
```

```
java.lang.Object  
|  
+-- java.lang.Throwable  
|  
+-- java.lang.Exception  
|  
+-- java.io.IOException  
|  
+-- org.dvb.dsmcc.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

Declaration

public class **LoadingAbortedEvent** extends `AsynchronousLoadingEvent`

```

java.lang.Object
|
+-- java.util.EventObject
    |
    +-- org.dvb.dsmcc.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

Declaration

```
public class MPEGDeliveryErrorEvent extends AsynchronousLoadingEvent
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
|   |  
|   +-- org.dvb.dsmcc.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

Declaration

```
public class MPEGDeliveryException extends DSMCCEException
```

```
java.lang.Object  
|  
+-- java.lang.Throwable  
|  
+-- java.lang.Exception  
|  
+-- java.io.IOException  
|  
+-- org.dvb.dsmcc.DSMCCEException  
|  
+-- 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

Declaration

```
public class NotEntitledEvent extends AsynchronousLoadingEvent
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
|   |  
|   +-- org.dvb.dsmcc.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

Declaration

public class **NotEntitledException** extends [DSMCCEException](#)

```

java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- java.io.IOException
            |
            +-- org.dvb.dsmcc.DSMCCEException
                |
                +-- 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

Declaration

public class **NothingToAbortException** extends `DSMCCEException`

```
java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- java.io.IOException
            |
            +-- org.dvb.dsmcc.DSMCCEException
                |
                +-- 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

Declaration

```
public class NotLoadedException extends DSMCCEException
```

```
java.lang.Object  
|  
+-- java.lang.Throwable  
|  
+-- java.lang.Exception  
|  
+-- java.io.IOException  
|  
+-- org.dvb.dsmcc.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

Declaration

```
public class NPTDiscontinuityEvent extends NPTStatusEvent
```

```
java.lang.Object
|
+-- java.util.EventObject
    |
    +-- org.dvb.dsmcc.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 the `NPTDiscontinuityEvent` 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 the `NPTDiscontinuityEvent` 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 the `NPTDiscontinuityEvent` is because of invalid data in a new `NPTReferenceDescriptor` then this is the time when that new descriptor was known to be invalid. If `NPTDiscontinuityEvent` 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. Where an NPT value is unknown or cannot be computed, -1 shall be used.

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

Declaration

```
public interface NPTListener extends java.util.EventListener
```

All Superinterfaces:

```
java.util.EventListener
```

Description

Objects that implement the `NPTListener` interface can receive `NPTStatusEvent` and `NPTRateChangeEvent` 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

`receiveRateChangeEvent(NPTRateChangeEvent)`

```
public void receiveRateChangeEvent(NPTRateChangeEvent e)
```

Send a `NPTRateChangeEvent` to a registered listener.

Parameters:

`e` - the `NPTRateChangeEvent` event.

org.dvb.dsmcc NPTPresentEvent

Declaration

```
public class NPTPresentEvent extends NPTStatusEvent
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
|   |  
|   +-- org.dvb.dsmcc.NPTStatusEvent  
|       |  
|       +-- org.dvb.dsmcc.NPTPresentEvent
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

Sent to listeners on a DSMCCStream object when NPT newly appears for that DSMCC stream when it was not previously present. This is specific to the particular timebase for this stream.

Since:

MHP 1.0.1

Constructors

NPTPresentEvent(DSMCCStream)

```
public NPTPresentEvent(DSMCCStream source)
```

Construct an event.

Parameters:

`source` - the DSMCCStream for which the NPT event appeared.

org.dvb.dsmcc NPTRate

Declaration

```
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

Declaration

```
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

Declaration

```
public class NPTRemovedEvent extends NPTStatusEvent
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
|   |  
|   +-- org.dvb.dsmcc.NPTStatusEvent  
|       |  
|       +-- org.dvb.dsmcc.NPTRemovedEvent
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

Sent to listeners on a DSMCCStream object when NPT which was present for that DSMCC stream is removed. This is specific to the particular timebase for this stream.

Since:

MHP 1.0.1

Constructors

NPTRemovedEvent(DSMCCStream)

```
public NPTRemovedEvent(DSMCCStream source)
```

Construct an event.

Parameters:

`source` - the DSMCCStream from which the NPT was removed

org.dvb.dsmcc NPTStatusEvent

Declaration

```
public abstract class NPTStatusEvent extends java.util.EventObject
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
|  
+-- org.dvb.dsmcc.NPTStatusEvent
```

All Implemented Interfaces:

```
java.io.Serializable
```

Direct Known Subclasses:

```
NPTDiscontinuityEvent, NPTPresentEvent, NPTRemovedEvent
```

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

Declaration

```
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 `DSMCCObject` 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

Declaration

```
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

Declaration

public class **ServerDeliveryErrorEvent** extends `AsynchronousLoadingEvent`

```
java.lang.Object
|
+-- java.util.EventObject
    |
    +-- org.dvb.dsmcc.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

Declaration

```
public class ServerDeliveryException extends DSMCCEException
```

```
java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- java.io.IOException
            |
            +-- org.dvb.dsmcc.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

Declaration

```
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)
    throws DSMCCEException, InterruptedException, InvalidAddressException, MPEGDeliveryException
```

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.

Calling this method on a `ServiceDomain` object already in the attached state shall imply a detach of the `ServiceDomain` object before the attach operation unless the `ServiceDomain` is already attached to the correct location. Hence if the attach operation fails, the appropriate exception for the failure mode shall be thrown and the `ServiceDomain` is left in a detached state and not attached to the former object carousel / service domain. If the `ServiceDomain` is already attached to the correct location then the method call shall have no effect.

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)
    throws DSMCCEException, InterruptedException, MPEGDeliveryException
```

This function is used to attach 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.

Calling this method on a `ServiceDomain` object already in the attached state shall imply a detach of the `ServiceDomain` object before the attach operation unless the `ServiceDomain` is already attached to the correct location. Hence if the attach operation fails, the appropriate exception for the failure mode shall be thrown and the `ServiceDomain` is left in a detached state and not attached to the former object carousel / service domain. If the `ServiceDomain` is already attached to the correct location then the method call shall have no effect.

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)
    throws ServiceXFREException, InterruptedException, MPEGDeliveryException
```

This function is used to attach 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.

Calling this method on a `ServiceDomain` object already in the attached state shall imply a detach of the `ServiceDomain` object before the attach operation unless the `ServiceDomain` is already attached to the correct location. Hence if the attach operation fails, the appropriate exception for the failure mode shall be thrown and the `ServiceDomain` is left in a detached state and not attached to the former object carousel / service domain. If the `ServiceDomain` is already attached to the correct location then the method call shall have no effect.

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. This exception shall not be thrown in this version of the specification.

detach()

```
public void detach()
    throws NotLoadedException
```

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 the attach was done with the `attach(locator, int)` signature, the locator is complemented with the `component_tag` value that the platform has identified during attaching the `ServiceDomain`. 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.

The syntax of the NSAP address is defined in section titled "LiteOptionsProfileBody" in annex B of the MHP specification. It contains the same fields as the locator syntax specified in the System integration aspects chapter. The locator is constructed by taking the fields out of the NSAP address and encoding them in the locator syntax together with the `component_tag` value that the platform has identified during attaching the `ServiceDomain`.

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 attached or null otherwise

Since:

MHP 1.0.1

getNSAPAddress()

```
public byte[] getNSAPAddress()  
    throws NotLoadedException
```

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)  
    throws NotLoadedException, InvalidLocatorException, FileNotFoundException
```

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:

l - 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

Declaration

```
public class ServiceXFRErrorEvent extends AsynchronousLoadingEvent
```

```
java.lang.Object
|
+-- java.util.EventObject
    |
    +-- org.dvb.dsmcc.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

Declaration

```
public class ServiceXFRException extends DSMCCException
```

```
java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- java.io.IOException
            |
            +-- org.dvb.dsmcc.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

Declaration

```
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

Declaration

```
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

Declaration

```
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

Declaration

```
public class SuccessEvent extends AsynchronousLoadingEvent
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
|   |  
|   +-- org.dvb.dsmcc.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

Declaration

public class **UnknownEventException** extends [DSMCCEException](#)

```

java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- java.io.IOException
            |
            +-- org.dvb.dsmcc.DSMCCEException
                |
                +-- 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

`DatagramSocketBufferControl` This class provides additional control over buffering for `DatagramSockets`.

org.dvb.net

DatagramSocketBufferControl

Declaration

```
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)
    throws SocketException
```

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)
    throws SocketException
```

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	
Interfaces	
<code>ConnectionListener</code>	This interface should be implemented by objects wishing to be notified about the connection status of a <code>ConnectionRCInterface</code> .
Classes	
<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.
Exceptions	
<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

Declaration

```
public class ConnectionEstablishedEvent extends ConnectionRCEvent
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
|   |  
|   +-- org.dvb.net.rc.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

Declaration

```
public class ConnectionFailedEvent extends ConnectionRCEvent
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
|   |  
|   +-- org.dvb.net.rc.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

Declaration

```
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

Declaration

```
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

Declaration

```
public class ConnectionRCEvent extends java.util.EventObject
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
|  
+-- org.dvb.net.rc.ConnectionRCEvent
```

All Implemented Interfaces:

```
java.io.Serializable
```

Direct Known Subclasses:

```
ConnectionEstablishedEvent, ConnectionFailedEvent,  
ConnectionTerminatedEvent
```

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

Declaration

```
public class ConnectionRCInterface extends RCInterface implements
    org.davic.resources.ResourceProxy
```

```
java.lang.Object
|
+--org.dvb.net.rc.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.

When a `ConnectionRCInterface` instance is first obtained by an application, the current target shall be set to the default. Applications which wish to use a non-default target need to set this target before attempting to reserve the `ConnectionRCInterface`. This is because if the application does not have the permission to use the default target, the `reserve()` method is required throw a `SecurityException`.

Constructors

ConnectionRCInterface()

```
protected ConnectionRCInterface()
```

Constructor for instances of this class. This constructor is provided for the use of implementations and specifications which extend this specification. Applications shall not define sub-classes of this class. Implementations are not required to behave correctly if any such application defined sub-classes are used.

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()  
    throws IOException, PermissionDeniedException
```

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()  
    throws PermissionDeniedException
```

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:

`getClient()` in interface `ConnectionRCInterface`

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()  
    throws IncompleteTargetException
```

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)
    throws PermissionDeniedException
```

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)
    throws IncompleteTargetException, PermissionDeniedException
```

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()
    throws PermissionDeniedException
```

Set the target for connections to the default.

Throws:

`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

Declaration

```
public class ConnectionTerminatedEvent extends ConnectionRCEvent
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
    |  
    +-- org.dvb.net.rc.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

Declaration

```
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

Declaration

```
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

Declaration

```
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.

TYPE_RCS

```
public static final int TYPE_RCS
```

Constant to indicate a DVB-RCS return channel.

Constructors

RCInterface()

```
protected RCInterface()
```

Constructor for instances of this class. This constructor is provided for the use of implementations and specifications which extend this specification. Applications shall not define sub-classes of this class. Implementations are not required to behave correctly if any such application defined sub-classes are used.

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. Note, applications wishing to discover whether a return channel interface is connection oriented or not are recommended to test whether an object is an instance of `ConnectionRCInterface` or not. A non-connection oriented interface really means a permanently connected return channel.

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

Declaration

```
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:

`addResourceStatusEventListener(ResourceStatusListener)` in interface `RCInterfaceManager`

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:

`removeResourceStatusEventListener(ResourceStatusListener)` in interface `RCInterfaceManager`

Parameters:

`listener` - the object which is no longer interested

org.dvb.net.rc RCInterfaceReleasedEvent

Declaration

```
public class RCInterfaceReleasedEvent extends org.davic.resources.ResourceStatusEvent
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
    |  
    +-- org.davic.resources.ResourceStatusEvent  
        |  
        +-- org.dvb.net.rc.RCInterfaceReleasedEvent
```

All Implemented Interfaces:

java.io.Serializable

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

Declaration

```
public class RCInterfaceReservedEvent extends org.davic.resources.ResourceStatusEvent
```

```
java.lang.Object  
|  
+-- java.util.EventObject  
|   |  
|   +-- org.davic.resources.ResourceStatusEvent  
|       |  
|       +-- org.dvb.net.rc.RCInterfaceReservedEvent
```

All Implemented Interfaces:

java.io.Serializable

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

Declaration

```
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

RCPermission(String, String)

```
public RCPermission(java.lang.String name, java.lang.String actions)
```

Creates a new RCPermission object with the specified name. The name is the symbolic name of the RCPermission, 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 `RCPermission`
`actions` - should be null.

Methods

implies(Permission)

```
public boolean implies(java.security.Permission p)
```

Checks if this `RCPermission` “implies” the specified `Permission`.

More specifically, this returns true if and only if:

- `p` is an instance of `RCPermission`, and
- `p`'s name is implied by the name of this permission, as described by the wildcarding rules specified in the the description of this class.

Overrides:

`java.security.BasicPermission.implies(java.security.Permission)` in class
`java.security.BasicPermission`

Parameters:

`p` - The `Permission` to check against.

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	
Interfaces	
<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>AppStateChangeListenerEventListener</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.
Classes	
<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>).
<code>RunningApplicationsFilter</code>	Instances of <code>RunningApplicationsFilter</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>).
Exceptions	
<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.

Class Summary

`LanguageNotAvailable-
Exception`

The `LanguageNotAvailableException` 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

Declaration

```
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.

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”. If no application names are signalled, an empty string shall be returned.

Returns:

the name of the application

Since:

MHP1.0

getName(String)

```
public java.lang.String getName(java.lang.String iso639code)
    throws LanguageNotAvailableException
```

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 or if the parameter passed is null

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. If no application names are signalled, an array of length zero shall be returned.

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 to 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 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 or null

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)
    throws IllegalProfileParameterException
```

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 or if the parameter passed in is null

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.

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

Declaration

```
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

Declaration

```
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

equals(Object)

```
public boolean equals(java.lang.Object obj)
```

Compares two AppIDs for equality.

Overrides:

java.lang.Object.equals(java.lang.Object) in class java.lang.Object

Parameters:

obj - the reference object with which to compare.

Returns:

true if this obj is an AppID and its Organisation ID and its Application ID match the ID's for this AppID; false otherwise.

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

hashCode()

```
public int hashCode()
```

Returns a hash code value for this AppID. The hashcode for two AppID's with the same Organisation ID and Application ID are equal.

Overrides:

java.lang.Object.hashCode() in class java.lang.Object

Returns:

a hash code value for this AppID

toString()

```
public java.lang.String toString()
```

This method returns a string containing the Hex representation of the 48 bit number. The string shall be formatted as specified in the section on "Text encoding of application identifiers" in the System Integration chapter of the MHP specification.

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

Declaration

```
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 calls are 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 under the following conditions:

- if the application (DVB-J or DVB-HTML) is in the not loaded or paused states,
- if a DVB-J application is in the “loaded” state,
- if a DVB-HTML application is in the “loading” state.

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 under the following conditions:

- if the application (DVB-J or DVB-HTML) is in the not loaded or paused states,
- if a DVB-J application is in the “loaded” state,
- if a DVB-HTML application is in the “loading” state.

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

Declaration

```
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 and only 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

org.dvb.application AppsDatabase

Declaration

```
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` or `RunningApplicationsFilter` filters may return a non empty Enumeration. If the filter object is not an instance of `CurrentServiceFilter` or `RunningApplicationsFilter` or a subclass of either 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` or `RunningApplicationsFilter` filters may return a non empty Enumeration. If the filter object is not an instance of `CurrentServiceFilter` or `RunningApplicationsFilter` or one of their 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

Declaration

```
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

Declaration

```
public interface AppsDatabaseEventListener extends java.util.EventListener
```

All Superinterfaces:

java.util.EventListener

Description

The `AppsDatabaseEventListener` 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

Declaration

```
public abstract class AppsDatabaseFilter

java.lang.Object
|
+--org.dvb.application.AppsDatabaseFilter
```

Direct Known Subclasses:

`CurrentServiceFilter`, `RunningApplicationsFilter`

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`.

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

Declaration

```
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. These events are only generated for running applications or formerly running applications on completion of a state transition into the DESTROYED state. 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()
```

If the `hasFailed` method returns false, then the application the listener is tracking is now in `toState`. If the `hasFailed` method returns true, then the `toState` is the state where the state transition was attempted to but the transition failed. 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 or actual 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

Declaration

```
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

Declaration

```
public class CurrentServiceFilter extends AppsDatabaseFilter
```

```
java.lang.Object
|
|--org.dvb.application.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 `getAppsIDs`).

A `CurrentServiceFilter` is used to indicate that only applications that signalled as part of the current service shall be returned by the `getAppsAttributes` and `getAppIDs` methods of `AppsDatabase`. Externally authorized applications in the AIT are not considered to be signalled as part of the current service for this filter. 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

Declaration

```
public interface DVBHTMLProxy extends AppProxy
```

All Superinterfaces:

[AppProxy](#)

Description

A DVBHTMLProxy Object is a proxy to a DVBHTML application.

Fields

KILLED

```
public static final int KILLED
```

The application is in the killed state.

Since:

MHP 1.0.2

LOADING

```
public static final int LOADING
```

The application is in the loading state.

Since:

MHP 1.0.2

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

Declaration

```
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 `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` 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

Declaration

```
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

Declaration

```
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

org.dvb.application RunningApplicationsFilter

Declaration

```
public class RunningApplicationsFilter extends AppsDatabaseFilter

java.lang.Object
|
|--org.dvb.application.AppsDatabaseFilter
|   |
|   |--org.dvb.application.RunningApplicationsFilter
```

Description

Instances of `RunningApplicationsFilter` are used to set a filter on the list of applications that are retrieved from the `AppsDatabase` (See methods `getAppsAttributes` and `getAppsIDs`).

A `RunningApplicationsFilter` is used to indicate that only applications that are running as part of the current service shall be returned by the `getAppsAttributes` and `getAppIDs` methods of `AppsDatabase`. Externally authorized applications in the AIT shall be returned if they are currently running in the same service context as the caller. Subclasses of `RunningApplicationsFilter` can override the `accept` method so as to implement their own filter criteria on the `AppID`'s values.

Since:

MHP 1.0

Constructors

`RunningApplicationsFilter()`

```
public RunningApplicationsFilter()
public Constructor of the RunningApplicationsFilter
```

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

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

Declaration

```
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.

Methods

implies(Permission)

```
public boolean implies(java.security.Permission p)
```

Checks if the specified permission is “implied” by this object.

Overrides:

java.security.BasicPermission.implies(java.security.Permission) in class
java.security.BasicPermission

Parameters:

`p` - the permission to check against.

Returns:

true if the passed permission is equal to or implied by this permission, false otherwise.

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

Declaration

```
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

Declaration

```
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

Methods

implies(Permission)

```
public boolean implies(java.security.Permission p)
```

Checks if the specified permission is “implied” by this object.

Since name and actions aren't used, the only check needed is whether p is also a TunerPermission.

Overrides:

java.security.BasicPermission.implies(java.security.Permission) in class
java.security.BasicPermission

Parameters:

p - the permission to check against.

Returns:

true if the passed permission is equal to or implied by this permission, false otherwise.

Annex U (normative): Extended graphics APIs

Package org.dvb.ui

Description

Provides extended graphics functionality.

Class Summary	
Interfaces	
<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> .
Classes	
<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.
Exceptions	
<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

Declaration

```
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 $A_r = 1.0f$
- 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:

$$cn = (A_s * A_r) * C_s * F_s + A_d * C_d * F_d$$

$$An = (A_s * A_r) * F_s + A_d * F_d$$

$$Cn = cn / An$$

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
```

`DVBAAlphaComposite` object that implements the opaque CLEAR rule with an alpha (A_r) 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 than 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 than 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
```

DVBAAlphaComposite object that implements the opaque DST_IN rule with an alpha (A_r) of 1.0f.

See Also:[DST_IN](#)

DstOut

```
public static final DVBAAlphaComposite DstOut
```

DVBAAlphaComposite object that implements the opaque DST_OUT rule with an alpha (Ar) of 1.0f.

See Also:[DST_OUT](#)

DstOver

```
public static final DVBAAlphaComposite DstOver
```

DVBAAlphaComposite object that implements the opaque DST_OVER rule with an alpha (Ar) of 1.0f.

See Also:[DST_OVER](#)

Src

```
public static final DVBAAlphaComposite Src
```

DVBAAlphaComposite 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 than 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 than 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 `DVBAlphaComposite` 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 `DVBAlphaComposite`. If this `DVBAlphaComposite` does not have an alpha value, 1.0 is returned.

Returns:

the alpha value of this `DVBAlphaComposite`.

getInstance(int)

```
public static DVBAlphaComposite getInstance(int rule)
```

Creates an `DVBAlphaComposite` object with the specified rule. The value for alpha shall be 1.0f.

Parameters:

`rule` - the compositing rule

Returns:

an `DVBAlphaComposite` object with the specified rule.

getInstance(int, float)

```
public static DVBAlphaComposite getInstance(int rule, float alpha)
```

Creates an `DVBAlphaComposite` object with the specified rule and the constant alpha (A_r) to multiply with the alpha of the source (A_s). The source is multiplied with the specified alpha before being composited with the destination.

Parameters:

`rule` - the compositing rule

`alpha` - the constant alpha (A_r) to be multiplied with the alpha of the source (A_s). `alpha` must be a floating point number in the inclusive range [0.0, 1.0].

Returns:

an `DVBAlphaComposite` 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 `DVBAlphaComposite`.

Returns:

the compositing rule of this `DVBAlphaComposite`.

org.dvb.ui DVBBufferedImage

Declaration

```
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; Instances of DVBBufferedImage shall be considered to be off-screen images for the purpose of the inherited method `Image.getGraphics`.

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. Calls to this method after calls to the `dispose` method on the same instance shall return null.

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. Calls to this method after calls to the `dispose` method on the same instance shall fail silently.

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`. Calls to this method after calls to the `dispose` method on the same instance shall return null.

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 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 which implement `java.awt.image.BufferedImage` 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`. Calls to this method after calls to the `dispose` method on the same instance shall return null.

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. Calls to this method after calls to the `dispose` method on the same instance shall return null.

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 or if the dispose method has been called on this instance

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 or if the dispose method has been called on this instance

Since:

MHP 1.0

getScaledInstance(int, int, int)

```
public java.awt.Image getScaledInstance(int width, int height, int hints)
```

Creates a scaled version of this image. A new `Image` object is returned which will render the image at the specified `width` and `height` by default. The new `Image` object may be loaded asynchronously even if the original source image has already been loaded completely. If either the `width` or `height` is a negative number then a value is substituted to maintain the aspect ratio of the original image dimensions. If the `dispose` method has been called on this instance than null shall be returned.

Overrides:

`java.awt.Image.getScaledInstance(int, int, int)` in class `java.awt.Image`

Parameters:

`width` - the width to which to scale the image.

`height` - the height to which to scale the image.

`hints` - flags to indicate the type of algorithm to use for image resampling.

Returns:

a scaled version of the image.

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.

If the `dispose` method has been called on this instance then null shall be returned. The source returned by this method is platform generated to provide access to the current contents of the `DVBBufferedImage` `buffer`.

See Also:

`java.awt.image.ImageProducer`

getSubimage(int, int, int, int)

```
public DVBBufferedImage getSubimage(int x, int y, int w, int h)
    throws DVBRasterFormatException
```

Returns a subimage defined by a specified rectangular region. The returned `DVBBufferedImage` shares the same data array as the original image. If the `dispose` method has been called on this instance then null shall be returned.

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 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. Calls to this method after calls to the `dispose` method on the same instance shall throw an `ArrayIndexOutOfBoundsException`.

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.

Calls to this method after calls to the `dispose` method on the same instance shall throw an `ArrayIndexOutOfBoundsException`.

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

Declaration

```
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. 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. Note: org.dvb.ui.DVBColor adds support for alpha (compared to JDK1.1.8) and is intended to be compatible with the JDK1.2 java.awt.Color class - since org.dvb.ui.DVBColor extends javax.tv.graphics.AlphaColor which in turn extends java.awt.Color. In implementations where java.awt.Color supports alpha, such as JDK1.2, etc., the alpha-related methods in org.dvb.ui.DVBColor could just call super.

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. If this color has no alpha value, alpha will be set to 255 (opaque).

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:

MHP 1.0

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:

MHP 1.0

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:

MHP 1.0

org.dvb.ui DVBGraphics

Declaration

```
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. In implementations which implement `java.awt.Graphics2D`, such as JDK1.2, etc., `org.dvb.ui.DVBGraphics` inherits from `java.awt.Graphics2D`. Otherwise, `org.dvb.ui.DVBGraphics` inherits from `java.awt.Graphics`. **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_OVER`. Note that the default rule of `SRC_OVER` may not give the highest performance. Under many circumstances, applications will find that the `SRC` rule will give higher performance. 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.SrcOver)`.
- Calling `setXORMode` on an instance of this class shall be equivalent to calling `setDVBComposite` with a special and implementation dependent `DVBAlphaComposite` object which 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 DVBBGraphics context. This method could delegate to a java.awt.Graphics2D object where available

Returns:

the current DVBBGraphics DVBAAlphaComposite, which defines a compositing style.

Since:

MHP 1.0

See Also:

[setDVBComposite\(DVBAAlphaComposite\)](#)

getType()

```
public int getType()
```

Returns the Sample Model (DVBBBufferedImage.TYPE_BASE, DVBBBufferedImage.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:

[DVBBBufferedImage](#)

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)
    throws UnsupportedOperationException
```

Sets the `DVBAlphaComposite` for the `DVBGraphics` context. The `DVBAlphaComposite` 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

Declaration

```
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 thrown.

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

Declaration

```
public class DVBTextLayoutManager implements org.havi.ui.HTextLayoutManager
```

```
java.lang.Object  
|  
+--org.dvb.ui.DVBTextLayoutManager
```

All Implemented Interfaces:

org.havi.ui.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 set by the setInsets method. These Insets are added to the ones passed to the `render` method for rendering the text. When not previously set, zero Insets are returned.

Returns:

Insets set by the setInsets method

getLetterSpace()

```
public int getLetterSpace()
```

Get the letter space setting. This is a 16 bit signed integer specifying in units of 1/256th point the required increase in the spacing between consecutive characters. It corresponds to the "track" parameter in the MHP text rendering rules.

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 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,  
    org.havi.ui.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 `DVBTextLayoutManager`

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. The setting shall be one of `HORIZONTAL_CENTER`, `HORIZONTAL_END_ALIGN` or `HORIZONTAL_START_ALIGN`. The failure mode if other values are used is implementation dependent.

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 which shall be used by this DVTextLayoutManager to provide a “virtual margin”. These shall be added to the insets passed to the `Render` method (which are to be considered as “bounds”). 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. This is a 16 bit signed integer specifying in units of 1/256th point the required increase in the spacing between consecutive characters. It corresponds to the “track” parameter in the MHP text rendering rules.

Parameters:

letterSpace - letter space setting

setLineOrientation(int)

```
public void setLineOrientation(int lineOrientation)
```

Set the line orientation. The setting shall be one of `LINE_ORIENTATION_VERTICAL`, `LINE_ORIENTATION_HORIZONTAL`. The failure mode if other values are used is implementation dependent.

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. The setting shall be one of `START_CORNER_UPPER_LEFT`, `START_CORNER_UPPER_RIGHT`, `START_CORNER_LOWER_LEFT` or `START_CORNER_LOWER_RIGHT`. The failure mode if other values are used is implementation dependent.

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. The setting shall be one of `VERTICAL_CENTER`, `VERTICAL_END_ALIGN` or `VERTICAL_START_ALIGN`. The failure mode if other values are used is implementation dependent.

Parameters:

`verticalAlign` - Vertical alignment setting

org.dvb.ui FontFactory

Declaration

```
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. The two constructors of this class allow fonts to be downloaded either through the font index file of the application or directly from a font file in the format(s) specified in the main body of the specification.

Constructors

FontFactory()

```
public FontFactory()
    throws FontFormatException, IOException
```

Constructs a **FontFactory** for the font index file bound to this application in the application signalling. 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.
- 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)
    throws IOException, FontFormatException
```

Constructs a **FontFactory** for the font file 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 location of the font file

Throws:

- 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)
    throws FontNotAvailableException, FontFormatException, IOException
```

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.

`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

Declaration

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

Declaration

```
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

Declaration

```
public interface TestOpacity
```

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

Declaration

```
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 as a result of a call to the render method. The timing of this is implementation dependent.

Methods

notifyTextOverflow(String, HVisible, boolean, boolean)

```
public void notifyTextOverflow(java.lang.String markedUpString, org.havi.ui.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

Declaration

```
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 `DST_IN` 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 : Void

Annex W (informative): DVB-J examples

W.1 DVB-J Application lifecycle implementation example

```
import org.havi.ui.HScene;
import org.havi.ui.HSceneFactory;
import java.awt.Component;
import java.awt.BorderLayout;
import org.dvb.ui.DVBColor;
import javax.tv.xlet.Xlet;
import javax.tv.xlet.XletContext;
import javax.tv.xlet.XletStateChangeException;

public class HelloDVB extends Component implements Xlet {

    private XletContext context;
    private HScene scene;

    public void initXlet(XletContext context) throws XletStateChangeException {
        this.context = context;
        scene = HSceneFactory.getInstance().getDefaultHScene();
        scene.setBounds( 90,72,540,432 );
        scene.setLayout(new BorderLayout(0, 0));
        scene.add(this, "Center");
    }

    public void startXlet() throws XletStateChangeException {
        scene.setVisible(true);
        requestFocus();
    }

    public void pauseXlet() {
        scene.setVisible(false);
    }

    public void destroyXlet(boolean unconditional) throws XletStateChangeException {
        scene.dispose();
    }

    public void paint(java.awt.Graphics g) {
        g.setColor(new DVBColor(0, 0, 70, 180));
        g.fillRect(0, 0, getSize().width, getSize().height);
        g.setColor(DVBColor.yellow);
        g.drawString("Hello DVB", (getSize().width-110) / 2, getSize().height / 2);
    }
}
```

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 implements 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(ctx, "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 = (String) ctx.getXletProperty("dvb.app.id");
        String orgId = (String) ctx.getXletProperty("dvb.org.id");

        Remote obj;
        try {
            org.dvb.io.ixc.IxcRegistry.lookup(ctx, "/" + orgId + "/" + appId + "/myserver");
            MyService r = (MyService) obj;
            System.out.println("Success " + r
                + ", " + r.getData());
        } catch (Exception ex) {
        }
    }
}
```

W.3 Example of use of video drip feed

```
import java.lang.*;
import java.io.*;
import javax.tv.xlet.*;
import javax.media.*;
import java.net.URL;
import org.dvb.media.DripFeedPermission;
import org.dvb.media.DripFeedDataSource;
```

```

/**
 * VideoDripTest creates an instance of DripFeedDataSource and creates a player
 * using this source.
 */
public class SingleVideoDripTest implements javax.tv.xlet.Xlet
{
    static final int    MAX_DRIP_DATA_SIZE = 32000;
    private DripFeedDataSource dripDataSource = null;
    private Player dripPlayer = null, oldPlayer = null;

    public void initXlet(XletContext ctx) throws XletStateChangeException
    {
        /* Check if this application has permission to play video drip
         * feed or not.
         */
        System.err.println("initXlet called");

        SecurityManager sm = System.getSecurityManager();

        System.err.println("Checking the permission");

        if (sm != null) {
            try {
                sm.checkPermission(new DripFeedPermission(""));
            }
            catch (java.lang.SecurityException ex) {
                throw new XletStateChangeException(ex.getMessage());
            }
        }

        System.err.println("Exiting initXlet method");
    }

    public void startXlet() throws XletStateChangeException
    {
        byte[] dripData = new byte[MAX_DRIP_DATA_SIZE];

        /* Assumption: No media player is active. If this is not true, we
         * need to get the current player and stop/close it.
         */
        System.err.println("startXlet called");

        try {
            /* Create a data source for video drip */
            dripDataSource = new DripFeedDataSource();

            System.err.println("Got the DataSource");

            /* Create a player and start it */
            dripPlayer = Manager.createPlayer(dripDataSource);
        }
        catch (IOException ioel)
        {
            System.err.println(ioel.getMessage());
            throw new XletStateChangeException(ioel.getMessage());
        }
        catch (NoPlayerException pse)
        {
            System.err.println(pse.getMessage());
            throw new XletStateChangeException(pse.getMessage());
        }

        System.err.println("Starting Drip Player");

        dripPlayer.start();

        System.err.println("Started the player");

        /* Read drip data from a file */
        try

```

```

    {
        FileInputStream fin = new FileInputStream("images.mpg");
        fin.read(dripData);
        fin.close();
    }
    catch (IOException ioe2)
    {
        System.err.println("IOException: " + ioe2.getMessage());
        throw new XletStateChangeException(ioe2.getMessage());
    }
    catch (SecurityException se)
    {
        System.err.println(se.getMessage());
        throw new XletStateChangeException("Security Exception" + se.getMessage());
    }
}

System.err.println("Feeding data to the datasource");

/* Feed the data to the data source */
dripDataSource.feed(dripData);

System.err.println("Fed data to the datasource");
}

public void pauseXlet()
{
}

public void destroyXlet(boolean unconditional)
{
    dripPlayer.close();
}
}

```

W.4 Example of CPU bound animation

```

/**
 * This is an example of doing CPU-bound animation. This
 * code attempts to do animation as fast as possible, using
 * a low-priority thread so that the animation doesn't interfere
 * with more important tasks, like responding to user input. Animation
 * is limited to 25 frames per second, just in case we're running on
 * a really fast box.
 */

import java.awt.Graphics;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Toolkit;
import org.dvb.ui.DVBBufferedImage;

public abstract class AnimatedView extends Component implements Runnable {

    private Thread worker = null;
    private Graphics theScreen = null;
    private DVBufferedImage buffer = null;
    private boolean stopping = false;

    /**
     * Called by the Xlet to start animation. Must never be called
     * when animation is in progress!
     */
    public synchronized void startAnimation() {
        if (worker != null) {
            throw new IllegalStateException();// It's a bug
        }
        worker = new Thread(this);
        worker.setPriority(2);// Animation is CPU-bound
        theScreen = getGraphics();// Component.getGraphics()
        Dimension d = getSize();
        buffer = new DVBufferedImage(d.width, d.height);
    }
}

```

```

stopping = false;
worker.start();
}

/**
 * Stop animation, and don't return until it has stopped.
 */
public synchronized void stopAnimation() {
if (worker == null) {
    throw new IllegalStateException();// it's a bug
}
stopping = true;
for (;;) { // Wait until stopped
    if (Thread.interrupted()) { // Xlet being terminated
        return;
    }
    notifyAll();
    try {
        wait();
    } catch (InterruptedException ex) {
        Thread.currentThread().interrupt();
    }
    if (!stopping) {
        theScreen.dispose();
        theScreen = null;
        buffer.flush();
        buffer = null;
        worker = null;
        return;
    }
}
}

/**
 * Run the animation until stopped. Called from the worker thread
 * only. This will probably be CPU-bound, as it sets an ambitious
 * target of 25fps animation.
 */
public void run() {

long start = System.currentTimeMillis();
long lastFrameTime = start - 40;
long now = 0;

animation:
for (;;) {

    // Wait until at least 1/25 of a second after last frame, and
    // bail out of thread if we're stopping
    synchronized(this) {
for (;;) {
    if (stopping) {
        stopping = false;
        notifyAll();
        return;
    }
    if (Thread.interrupted()) { // Xlet being terminated
        return;
    }
    now = System.currentTimeMillis();
    long delta = now - lastFrameTime;
    if (delta >= 40) {
        break;
    } else {
        try {
            wait(40 - delta);
        } catch (InterruptedException ex) {
            Thread.currentThread().interrupt();
        }
    }
}
}
}
}

```



```

    }
    lastFrameTime = now;
    long timeSinceStart = now - start;

    Graphics g = buffer.getGraphics();

    for (int part = 0; part < getNumParts(); part++) {
        drawPart(part, g, timeSinceStart);

        if (shouldStop()) {
            continue animation;// Bail out if animation should stop
        }
    }

    g.dispose();
    theScreen.drawImage(buffer, 0, 0, null);
    Toolkit.getDefaultToolkit().sync();
}
}

private synchronized boolean shouldStop() {
return Thread.interrupted() || stopping;
}

public synchronized void paint(Graphics g) {
if (worker == null || stopping) {
    // Paint whatever we paint when we're not animating
} else {
    // Probably do nothing.  If our animation target were
    // significantly less than 25 fps, we'd want to set a variable
    // to cause a repaint ASAP, then call notifyAll() to break
    // the animation loop out of any wait() it might be in.
}
}

/**
 * Draw a part of the animation. For each frame, AnimatedView will call
 * this method first for part 0, then part 1, up to getNumParts()-1.
 * Between each part, AnimatedView will check if animation needs to
 * stop for some reason.
 * <p>
 * Subclasses should ensure that drawing the entire scene is divided
 * into enough parts to ensure that animation can be stopped quickly.
 *
 * @see #getNumParts
 */
protected abstract void drawPart(int num, Graphics g, long timeSinceStart);

/**
 * @return the number of parts in this animation. This determines how
 * many times drawPart will be called.
 *
 * @see #drawPart
 */
protected abstract int getNumParts();
}

```

Annex X (normative): Test support

Package

Broadcast model

Description

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 DVBTTest.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 DVBTTest.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.

Class Summary	
Classes	
<code>DVBTest</code>	The DVBTest 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

Declaration

```
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)
    throws IOException
```

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)
    throws IOException
```

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 DVCTest 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)
    throws IOException
```

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 DVCTest 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)
    throws IOException
```

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 DVBTest 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 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

Declaration

```
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 each 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)
    throws AlreadyBoundException
```

Exports an 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

Throws:

`AlreadyBoundException` - if this Xlet has previously exported an object under the given name.

`NullPointerException` - if `xc`, `name` or `obj` is null

list(XletContext)

```
public static java.lang.String[] list(javax.tv.xlet.XletContext xc)
```

Returns an array of string path 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 `lxcRegistry.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)
    throws NotBoundException, RemoteException
```

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 application 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 each 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 path is not currently bound.

`RemoteException` - If a remote stub class cannot be generated for the object being imported.

`java.lang.IllegalArgumentException` - If the path 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.

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

unbind(XletContext, String)

```
public static void unbind(javax.tv.xlet.XletContext xc, java.lang.String name)
    throws NotBoundException
```

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, all real-time media components sharing the same clock are presented by the same JMF player. 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.

Index

A

- abort()
 - of org.dvb.dsmcc.DSMCCObject 525
- accept(AppID)
 - of org.dvb.application.AppsDatabaseFilter 633
 - of org.dvb.application.CurrentServiceFilter 637
 - of org.dvb.application.RunningApplicationsFilter 644
- ActiveFormatDescriptionChangedEvent
 - of org.dvb.media 468
- ActiveFormatDescriptionChangedEvent(Object, int)
 - of org.dvb.media.ActiveFormatDescriptionChangedEvent 468
- add(int, String)
 - of org.dvb.user.Preference 372
- add(String)
 - of org.dvb.user.Preference 372
- add(String[])
 - of org.dvb.user.Preference 372
- addAllArrowKeys()
 - of org.dvb.event.UserEventRepository 356
- addAllColourKeys()
 - of org.dvb.event.UserEventRepository 356
- addAllNumericKeys()
 - of org.dvb.event.UserEventRepository 356
- addAppStateChangeListener(AppStateChangeListener)
 - of org.dvb.application.AppProxy 620
- addBouquetMonitoringListener(SIMonitoringListener, int)
 - of org.dvb.si.SIDatabase 399
- addConnectionListener(ConnectionListener)
 - of org.dvb.net.rc.ConnectionRCInterface 593
- addEventPresentFollowingMonitoringListener(SIMonitoringListener, int, int, int)
 - of org.dvb.si.SIDatabase 400
- addEventScheduleMonitoringListener(SIMonitoringListener, int, int, int, Date, Date)
 - of org.dvb.si.SIDatabase 400
- addExclusiveAccessToAWTEvent(ResourceClient, UserEventRepository)
 - of org.dvb.event.EventManager 343
- addKey(int)
 - of org.dvb.event.UserEventRepository 356
- addListener(AppsDatabaseEventListener)
 - of org.dvb.application.AppsDatabase 626
- addNetworkMonitoringListener(SIMonitoringListener, int)
 - of org.dvb.si.SIDatabase 401
- addNPTListener(NPTListener)
 - of org.dvb.dsmcc.DSMCCStream 532
- addObjectChangeListener(ObjectChangeListener)
 - of org.dvb.dsmcc.DSMCCObject 526
- addPMTServiceMonitoringListener(SIMonitoringListener, int, int, int)
 - of org.dvb.si.SIDatabase 402
- addResourceStatusEventListener(ResourceStatusListener)
 - of org.dvb.event.EventManager 344
 - of org.dvb.net.rc.RCInterfaceManager 602
- addServiceMonitoringListener(SIMonitoringListener, int, int)

- of org.dvb.si.SIDatabase 402
- addSubtitleListener(SubtitleListener)
 - of org.dvb.media.SubtitlingEventControl 494
- addTextOverflowListener(TextOverflowListener)
 - of org.dvb.ui.DVBTextLayoutManager 679
- addUserEvent(UserEvent)
 - of org.dvb.event.UserEventRepository 356
- addUserEventListener(UserEventListener, ResourceClient, UserEventRepository)
 - of org.dvb.event.EventManager 344
- addUserEventListener(UserEventListener, UserEventRepository)
 - of org.dvb.event.EventManager 345
- addUserPreferenceChangeListener(UserPreferenceChangeListener)
 - of org.dvb.user.UserPreferenceManager 377
- addVideoFormatListener(VideoFormatListener)
 - of org.dvb.media.VideoFormatControl 498
- AFD_14_9
 - of org.dvb.media.VideoFormatControl 495
- AFD_14_9_TOP
 - of org.dvb.media.VideoFormatControl 495
- AFD_16_9
 - of org.dvb.media.VideoFormatControl 495
- AFD_16_9_SP_14_9
 - of org.dvb.media.VideoFormatControl 495
- AFD_16_9_SP_4_3
 - of org.dvb.media.VideoFormatControl 495
- AFD_16_9_TOP
 - of org.dvb.media.VideoFormatControl 495
- AFD_4_3
 - of org.dvb.media.VideoFormatControl 496
- AFD_4_3_SP_14_9
 - of org.dvb.media.VideoFormatControl 496
- AFD_GT_16_9
 - of org.dvb.media.VideoFormatControl 496
- AFD_NOT_PRESENT
 - of org.dvb.media.VideoFormatControl 496
- AFD_SAME
 - of org.dvb.media.VideoFormatControl 496
- APP_ADDED
 - of org.dvb.application.AppsDatabaseEvent 629
- APP_CHANGED
 - of org.dvb.application.AppsDatabaseEvent 629
- APP_DELETED
 - of org.dvb.application.AppsDatabaseEvent 629
- AppAttributes
 - of org.dvb.application 611
- AppIcon
 - of org.dvb.application 616
- AppIcon()
 - of org.dvb.application.AppIcon 616
- AppID
 - of org.dvb.application 617
- AppID(int, int)
 - of org.dvb.application.AppID 617
- AppProxy
 - of org.dvb.application 619

AppsControlPermission
of org.dvb.application 623

AppsControlPermission()
of org.dvb.application.AppsControlPermission 623

AppsControlPermission(String, String)
of org.dvb.application.AppsControlPermission 623

AppsDatabase
of org.dvb.application 625

AppsDatabaseEvent
of org.dvb.application 629

AppsDatabaseEvent(int, AppID, Object)
of org.dvb.application.AppsDatabaseEvent 630

AppsDatabaseEventListener
of org.dvb.application 631

AppsDatabaseFilter
of org.dvb.application 633

AppsDatabaseFilter()
of org.dvb.application.AppsDatabaseFilter 633

AppStateChangeEvent
of org.dvb.application 634

AppStateChangeEvent(AppID, int, int, Object, boolean)
of org.dvb.application.AppStateChangeEvent 634

AppStateChangeListener
of org.dvb.application 636

ASPECT_RATIO_16_9
of org.dvb.media.VideoFormatControl 496

ASPECT_RATIO_2_21_1
of org.dvb.media.VideoFormatControl 496

ASPECT_RATIO_4_3
of org.dvb.media.VideoFormatControl 496

ASPECT_RATIO_UNKNOWN
of org.dvb.media.VideoFormatControl 496

AspectRatioChangedEvent
of org.dvb.media 469

AspectRatioChangedEvent(Object, int)
of org.dvb.media.AspectRatioChangedEvent 469

asynchronousLoad(AsynchronousLoadingEventListener)
of org.dvb.dsmcc.DSMCCObject 526

AsynchronousLoadingEvent
of org.dvb.dsmcc 521

AsynchronousLoadingEvent(DSMCCObject)
of org.dvb.dsmcc.AsynchronousLoadingEvent 521

AsynchronousLoadingEventListener
of org.dvb.dsmcc 522

attach(byte[])
of org.dvb.dsmcc.ServiceDomain 567

attach(Locator)
of org.dvb.dsmcc.ServiceDomain 567

attach(Locator, int)
of org.dvb.dsmcc.ServiceDomain 567

B

BackgroundVideoPresentationControl
of org.dvb.media 470

bind(XletContext, String, Remote)
 of org.dvb.io.ixc.IxcRegistry 709

BOUQUET
 of org.dvb.si.SIMonitoringType 433

BOUQUET_NAME
 of org.dvb.si.DescriptorTag 386

brighter()
 of org.dvb.ui.DVBColor 669

C

CA_FAILURE
 of org.dvb.media.PresentationChangedEvent 483

CA_IDENTIFIER
 of org.dvb.si.DescriptorTag 386

CA_RETURNED
 of org.dvb.media.PresentationChangedEvent 483

CABLE_DELIVERY_SYSTEM
 of org.dvb.si.DescriptorTag 386

CAException
 of org.dvb.media 471

CAException()
 of org.dvb.media.CAException 471

CAException(String)
 of org.dvb.media.CAException 471

cancelRequest()
 of org.dvb.si.SIRequest 440

CAPermission
 of org.dvb.net.ca 647

CAPermission(String)
 of org.dvb.net.ca.CAPermission 647

CAPermission(String, String)
 of org.dvb.net.ca.CAPermission 648

CAStopEvent
 of org.dvb.media 472

CAStopEvent(Controller)
 of org.dvb.media.CAStopEvent 472

CAStopEvent(Controller, int, int, int, MediaLocator)
 of org.dvb.media.CAStopEvent 472

CLEAR
 of org.dvb.ui.DVBAlphaComposite 656

Clear
 of org.dvb.ui.DVBAlphaComposite 655

close() - of org.davic.net.ca.MMIObjct 243

COMPONENT
 of org.dvb.si.DescriptorTag 386

connect()
 of org.dvb.media.DripFeedDataSource 476
 of org.dvb.net.rc.ConnectionRCInterface 594

connectionChanged(ConnectionRCEvent)
 of org.dvb.net.rc.ConnectionListener 589

ConnectionEstablishedEvent
 of org.dvb.net.rc 587

ConnectionEstablishedEvent(Object)
 of org.dvb.net.rc.ConnectionEstablishedEvent 587

ConnectionFailedEvent
 of org.dvb.net.rc 588
 ConnectionFailedEvent(Object)
 of org.dvb.net.rc.ConnectionFailedEvent 588
 ConnectionListener
 of org.dvb.net.rc 589
 ConnectionParameters
 of org.dvb.net.rc 590
 ConnectionParameters(String, String, String)
 of org.dvb.net.rc.ConnectionParameters 590
 ConnectionParameters(String, String, String, InetAddress[])
 of org.dvb.net.rc.ConnectionParameters 590
 ConnectionRCEvent
 of org.dvb.net.rc 592
 ConnectionRCEvent(Object)
 of org.dvb.net.rc.ConnectionRCEvent 592
 ConnectionRCInterface
 of org.dvb.net.rc 593
 ConnectionRCInterface()
 of org.dvb.net.rc.ConnectionRCInterface 593
 ConnectionTerminatedEvent
 of org.dvb.net.rc 597
 ConnectionTerminatedEvent(Object)
 of org.dvb.net.rc.ConnectionTerminatedEvent 597
 CONTENT
 of org.dvb.si.DescriptorTag 386
 convertSISStringToJavaString(byte[], int, int, boolean)
 of org.dvb.si.SIUtil 463
 COUNTRY_AVAILABILITY
 of org.dvb.si.DescriptorTag 386
 createFont(String, int, int)
 of org.dvb.ui.FontFactory 685
 createGraphics()
 of org.dvb.ui.DVBBufferedImage 661
 CurrentServiceFilter
 of org.dvb.application 637
 CurrentServiceFilter()
 of org.dvb.application.CurrentServiceFilter 637

D

D_D2_MAC
 of org.dvb.si.SIServiceType 452
 DAR_16_9
 of org.dvb.media.VideoFormatControl 497
 DAR_4_3
 of org.dvb.media.VideoFormatControl 497
 darker()
 of org.dvb.ui.DVBColor 670
 DATA_BROADCAST
 of org.dvb.si.DescriptorTag 387
 of org.dvb.si.SIServiceType 452
 DatagramSocketBufferControl
 of org.dvb.net 584
 Descriptor

- of org.dvb.si 384
- DescriptorTag
 - of org.dvb.si 386
- DESTROYED
 - of org.dvb.application.AppProxy 619
- detach()
 - of org.dvb.dsmcc.ServiceDomain 568
- DFC_PLATFORM
 - of org.dvb.media.VideoFormatControl 497
- DFC_PROCESSING_CCO
 - of org.dvb.media.VideoFormatControl 497
- DFC_PROCESSING_FULL
 - of org.dvb.media.VideoFormatControl 497
- DFC_PROCESSING_LB_14_9
 - of org.dvb.media.VideoFormatControl 497
- DFC_PROCESSING_LB_16_9
 - of org.dvb.media.VideoFormatControl 497
- DFC_PROCESSING_LB_2_21_1_ON_16_9
 - of org.dvb.media.VideoFormatControl 497
- DFC_PROCESSING_LB_2_21_1_ON_4_3
 - of org.dvb.media.VideoFormatControl 497
- DFC_PROCESSING_NONE
 - of org.dvb.media.VideoFormatControl 498
- DFC_PROCESSING_PAN_SCAN
 - of org.dvb.media.VideoFormatControl 498
- DFC_PROCESSING_UNKNOWN
 - of org.dvb.media.VideoFormatControl 498
- DFCChangedEvent
 - of org.dvb.media 474
- DFCChangedEvent(Object, int)
 - of org.dvb.media.DFCChangedEvent 474
- DIGITAL_RADIO_SOUND
 - of org.dvb.si.SIServiceType 452
- DIGITAL_TELEVISION
 - of org.dvb.si.SIServiceType 452
- disconnect()
 - of org.dvb.media.DripFeedDataSource 476
 - of org.dvb.net.rc.ConnectionRCInterface 594
- dispose()
 - of org.dvb.ui.DVBBufferedImage 661
- DripFeedDataSource
 - of org.dvb.media 475
- DripFeedDataSource()
 - of org.dvb.media.DripFeedDataSource 475
- DripFeedPermission
 - of org.dvb.media 478
- DripFeedPermission(String)
 - of org.dvb.media.DripFeedPermission 478
- DripFeedPermission(String, String)
 - of org.dvb.media.DripFeedPermission 478
- DSMCCEException
 - of org.dvb.dsmcc 523
- DSMCCEException()
 - of org.dvb.dsmcc.DSMCCEException 523
- DSMCCEException(String)

of org.dvb.dsmcc.DSMCCException 523
 DSMCCObject
 of org.dvb.dsmcc 524
 DSMCCObject(DSMCCObject, String)
 of org.dvb.dsmcc.DSMCCObject 525
 DSMCCObject(String)
 of org.dvb.dsmcc.DSMCCObject 525
 DSMCCObject(String, String)
 of org.dvb.dsmcc.DSMCCObject 525
 DSMCCStream
 of org.dvb.dsmcc 531
 DSMCCStream(DSMCCObject)
 of org.dvb.dsmcc.DSMCCStream 531
 DSMCCStream(String)
 of org.dvb.dsmcc.DSMCCStream 531
 DSMCCStream(String, String)
 of org.dvb.dsmcc.DSMCCStream 532
 DSMCCStreamEvent
 of org.dvb.dsmcc 535
 DSMCCStreamEvent(DSMCCObject)
 of org.dvb.dsmcc.DSMCCStreamEvent 535
 DSMCCStreamEvent(String)
 of org.dvb.dsmcc.DSMCCStreamEvent 535
 DSMCCStreamEvent(String, String)
 of org.dvb.dsmcc.DSMCCStreamEvent 536
 DST_IN
 of org.dvb.ui.DVBAlphaComposite 656
 DST_OUT
 of org.dvb.ui.DVBAlphaComposite 656
 DST_OVER
 of org.dvb.ui.DVBAlphaComposite 656
 DstIn
 of org.dvb.ui.DVBAlphaComposite 656
 DstOut
 of org.dvb.ui.DVBAlphaComposite 657
 DstOver
 of org.dvb.ui.DVBAlphaComposite 657
 DVB_HTML_application
 of org.dvb.application.AppAttributes 611
 DVB_J_application
 of org.dvb.application.AppAttributes 611
 DVBAlphaComposite
 of org.dvb.ui 655
 DVBBufferedImage
 of org.dvb.ui 660
 DVBBufferedImage(int, int)
 of org.dvb.ui.DVBBufferedImage 661
 DVBBufferedImage(int, int, int)
 of org.dvb.ui.DVBBufferedImage 661
 DVBClassLoader
 of org.dvb.lang 336
 DVBClassLoader(URL[])
 of org.dvb.lang.DVBClassLoader 336
 DVBClassLoader(URL[], ClassLoader)
 of org.dvb.lang.DVBClassLoader 336

DVBColor
 of org.dvb.ui 668
 DVBColor(Color)
 of org.dvb.ui.DVBColor 668
 DVBColor(float, float, float, float)
 of org.dvb.ui.DVBColor 668
 DVBColor(int, boolean)
 of org.dvb.ui.DVBColor 669
 DVBColor(int, int, int, int)
 of org.dvb.ui.DVBColor 669
 DVBGraphics
 of org.dvb.ui 672
 DVBGraphics()
 of org.dvb.ui.DVBGraphics 672
 DVBHTMLProxy
 of org.dvb.application 638
 DVBJProxy
 of org.dvb.application 640
 DVBMediaSelectControl
 of org.dvb.media 480
 DvbNetworkInterfaceSIUtil
 of org.dvb.net.tuning 650
 DVBRasterFormatException
 of org.dvb.ui 676
 DVBRasterFormatException(String)
 of org.dvb.ui.DVBRasterFormatException 676
 DVBTest
 of org.dvb.test 701
 DVBTextLayoutManager
 of org.dvb.ui 677
 DVBTextLayoutManager()
 of org.dvb.ui.DVBTextLayoutManager 679
 DVBTextLayoutManager(int, int, int, int, boolean, int, int, int)
 of org.dvb.ui.DVBTextLayoutManager 679

E

entryAdded(AppsDatabaseEvent)
 of org.dvb.application.AppsDatabaseEventListener 631
 entryChanged(AppsDatabaseEvent)
 of org.dvb.application.AppsDatabaseEventListener 631
 entryRemoved(AppsDatabaseEvent)
 of org.dvb.application.AppsDatabaseEventListener 631
 equals(Object)
 of org.dvb.application.AppID 617
 of org.dvb.application.AppsControlPermission 623
 of org.dvb.ui.DVBAlphaComposite 658
 of org.dvb.ui.DVBColor 670
 EventManager
 of org.dvb.event 343
 EventManager()
 of org.dvb.event.EventManager 343
 EXTENDED_EVENT
 of org.dvb.si.DescriptorTag 387

F

- Facility
 - of org.dvb.user 369
- Facility(String, String)
 - of org.dvb.user.Facility 369
- Facility(String, String[])
 - of org.dvb.user.Facility 369
- FAIL
 - of org.dvb.test.DVBTest 701
- feed(byte[])
 - of org.dvb.media.DripFeedDataSource 476
- FileAccessPermissions
 - of org.dvb.io.persistent 362
- FileAccessPermissions(boolean, boolean, boolean, boolean, boolean)
 - of org.dvb.io.persistent.FileAccessPermissions 362
- FileAttributes
 - of org.dvb.io.persistent 364
- findClass(String)
 - of org.dvb.lang.DVBClassLoader 337
- flush()
 - of org.dvb.ui.DVBBufferedImage 662
- FM_RADIO
 - of org.dvb.si.SIServiceType 452
- FontFactory
 - of org.dvb.ui 684
- FontFactory()
 - of org.dvb.ui.FontFactory 684
- FontFactory(URL)
 - of org.dvb.ui.FontFactory 684
- FontFormatException
 - of org.dvb.ui 686
- FontFormatException()
 - of org.dvb.ui.FontFormatException 686
- FontFormatException(String)
 - of org.dvb.ui.FontFormatException 686
- FontNotAvailableException
 - of org.dvb.ui 687
- FontNotAvailableException()
 - of org.dvb.ui.FontNotAvailableException 687
- FontNotAvailableException(String)
 - of org.dvb.ui.FontNotAvailableException 687
- FREQUENCY_LIST
 - of org.dvb.si.DescriptorTag 387
- FROM_CACHE
 - of org.dvb.dsmcc.DSMCCObject 524
- FROM_CACHE_ONLY
 - of org.dvb.si.SIInformation 422
- FROM_CACHE_OR_STREAM
 - of org.dvb.dsmcc.DSMCCObject 524
 - of org.dvb.si.SIInformation 422
- FROM_STREAM_ONLY
 - of org.dvb.dsmcc.DSMCCObject 525
 - of org.dvb.si.SIInformation 422
- fromActual()

of org.dvb.si.SIInformation 423

G

GeneralPreference

of org.dvb.user 370

GeneralPreference(String)

of org.dvb.user.GeneralPreference 370

getActions()

of org.dvb.application.AppsControlPermission 624

getActiveFormatDefinition()

of org.dvb.media.VideoFormatControl 498

getActiveVideoArea()

of org.dvb.media.VideoPresentationControl 503

getActiveVideoAreaOnScreen()

of org.dvb.media.VideoPresentationControl 503

getAID()

of org.dvb.application.AppID 617

getAlpha()

of org.dvb.ui.DVBAlphaComposite 659

of org.dvb.ui.DVBColor 670

getAppAttributes(AppID)

of org.dvb.application.AppsDatabase 626

getAppAttributes(AppsDatabaseFilter)

of org.dvb.application.AppsDatabase 626

getAppData()

of org.dvb.si.SIRetrievalEvent 442

getAppIcon()

of org.dvb.application.AppAttributes 611

getAppID()

of org.dvb.application.AppsDatabaseEvent 630

of org.dvb.application.AppStateChangeEvent 635

getAppIDs(AppsDatabaseFilter)

of org.dvb.application.AppsDatabase 627

getAppProxy(AppID)

of org.dvb.application.AppsDatabase 627

getAppsDatabase()

of org.dvb.application.AppsDatabase 627

getAspectRatio()

of org.dvb.media.VideoFormatControl 498

getAvailableCompositeRules()

of org.dvb.ui.DVBGraphics 673

getBestColorMatch(Color)

of org.dvb.ui.DVBGraphics 673

getBouquetID()

of org.dvb.si.SIBouquet 395

of org.dvb.si.SIMonitoringEvent 430

of org.dvb.si.SITransportStreamBAT 460

getByteAt(int)

of org.dvb.si.Descriptor 384

getCarouselId()

of org.dvb.dsmcc.ServiceXFRReference 575

getClient()

of org.dvb.event.RepositoryDescriptor 347

of org.dvb.net.rc.ConnectionRCInterface 594

- getClipRegion()
 - of org.dvb.media.VideoPresentationControl 503
 - of org.dvb.media.VideoTransformation 507
- getClosestMatch(VideoTransformation)
 - of org.dvb.media.BackgroundVideoPresentationControl 470
- getCode()
 - of org.dvb.event.UserEvent 349
- getColor()
 - of org.dvb.ui.DVBGraphics 673
- getComponentTag()
 - of org.dvb.si.PMTElementaryStream 390
- getConnectedTime()
 - of org.dvb.net.rc.ConnectionRCInterface 594
- getContent()
 - of org.dvb.si.Descriptor 384
- getContentLength()
 - of org.dvb.si.Descriptor 384
- getContentNibbles()
 - of org.dvb.si.SIEvent 416
- getContentType()
 - of org.dvb.media.DripFeedDataSource 476
- getControl(String)
 - of org.dvb.media.DripFeedDataSource 476
- getControls()
 - of org.dvb.media.DripFeedDataSource 477
- getCurrentTarget()
 - of org.dvb.net.rc.ConnectionRCInterface 594
- getDataRate()
 - of org.dvb.net.rc.RCInterface 601
- getDataSource()
 - of org.dvb.si.SIInformation 423
- getDecoderFormatConversion()
 - of org.dvb.media.VideoFormatControl 499
- getDenominator()
 - of org.dvb.dsmcc.NPTRate 556
- getDescriptorTags()
 - of org.dvb.si.SIBouquet 395
 - of org.dvb.si.SIInformation 423
 - of org.dvb.si.SINetwork 434
- getDisplayAspectRatio()
 - of org.dvb.media.VideoFormatControl 499
- getDNSServer()
 - of org.dvb.net.rc.ConnectionParameters 590
- getDuration()
 - of org.dvb.dsmcc.DSMCCStream 532
 - of org.dvb.media.DripFeedDataSource 477
 - of org.dvb.si.SIEvent 416
- getDVBComposite()
 - of org.dvb.ui.DVBGraphics 674
- getDvbLocator()
 - of org.dvb.si.PMTElementaryStream 390
 - of org.dvb.si.PMTService 392
 - of org.dvb.si.SIEvent 416
 - of org.dvb.si.SIService 446
 - of org.dvb.si.SITransportStream 458

getEITPresentFollowingFlag()
 of org.dvb.si.SIService 446
 getEITScheduleFlag()
 of org.dvb.si.SIService 446
 getElementaryPID()
 of org.dvb.si.PMTElementaryStream 390
 getElementaryStreams() - of org.davic.mpeg.NotAuthorizedException 230
 getEndTime()
 of org.dvb.si.SIMonitoringEvent 430
 getEventData()
 of org.dvb.dsmcc.StreamEvent 577
 getEventID()
 of org.dvb.si.SIEvent 417
 getEventId()
 of org.dvb.application.AppsDatabaseEvent 630
 of org.dvb.dsmcc.StreamEvent 578
 getEventList()
 of org.dvb.dsmcc.DSMCCStreamEvent 536
 getEventName()
 of org.dvb.dsmcc.StreamEvent 578
 getEventNPT()
 of org.dvb.dsmcc.StreamEvent 578
 getExpirationDate()
 of org.dvb.io.persistent.FileAttributes 364
 getFamily()
 of org.dvb.event.UserEvent 349
 getFavourites()
 of org.dvb.user.Preference 372
 getFileAttributes(File)
 of org.dvb.io.persistent.FileAttributes 364
 getFirstNPT()
 of org.dvb.dsmcc.NPTDiscontinuityEvent 553
 getFreeCAMode()
 of org.dvb.si.SIEvent 417
 of org.dvb.si.SIService 446
 getFromState()
 of org.dvb.application.AppStateChangeEvent 635
 getGraphics()
 of org.dvb.ui.DVBBufferedImage 662
 getHeight()
 of org.dvb.ui.DVBBufferedImage 662
 getHeight(ImageObserver)
 of org.dvb.ui.DVBBufferedImage 662
 getHorizontalAlign()
 of org.dvb.ui.DVBTextLayoutManager 679
 getHorizontalScalingFactors()
 of org.dvb.media.VideoPresentationControl 504
 getHorizontalTabSpacing()
 of org.dvb.ui.DVBTextLayoutManager 679
 getIconFlags()
 of org.dvb.application.AppIcon 616
 getIdentifier()
 of org.dvb.application.AppAttributes 611
 getImage()
 of org.dvb.ui.DVBBufferedImage 663

getInputVideoSize()
of org.dvb.media.VideoPresentationControl 504

getInsets()
of org.dvb.ui.DVBTextLayoutManager 680

getInstance()
of org.dvb.event.EventManager 345
of org.dvb.net.rc.RCInterfaceManager 602
of org.dvb.user.UserPreferenceManager 377

getInstance(int)
of org.dvb.ui.DVBAlphaComposite 659

getInstance(int, float)
of org.dvb.ui.DVBAlphaComposite 659

getInterface(InetAddress)
of org.dvb.net.rc.RCInterfaceManager 602

getInterface(Socket)
of org.dvb.net.rc.RCInterfaceManager 603

getInterface(URLConnection)
of org.dvb.net.rc.RCInterfaceManager 603

getInterfaces()
of org.dvb.net.rc.RCInterfaceManager 603

getIsServiceBound()
of org.dvb.application.AppAttributes 612

getKeyChar()
of org.dvb.event.UserEvent 349

getLastNPT()
of org.dvb.dsmcc.NPTDiscontinuityEvent 553

getLetterSpace()
of org.dvb.ui.DVBTextLayoutManager 680

getLevel1ContentNibbles()
of org.dvb.si.SIEvent 417

getLineOrientation()
of org.dvb.ui.DVBTextLayoutManager 680

getLineSpace()
of org.dvb.ui.DVBTextLayoutManager 680

getLocator()
of org.dvb.application.AppIcon 616
of org.dvb.dsmcc.ServiceDomain 568
of org.dvb.dsmcc.ServiceXFRReference 576

getModifiers()
of org.dvb.event.UserEvent 350

getMostFavourite()
of org.dvb.user.Preference 372

getMountPoint()
of org.dvb.dsmcc.ServiceDomain 569

getName()
of org.dvb.application.AppAttributes 612
of org.dvb.event.RepositoryDescriptor 347
of org.dvb.si.SIBouquet 395
of org.dvb.si.SIEvent 417
of org.dvb.si.SINetwork 434
of org.dvb.si.SIService 447
of org.dvb.user.Preference 372
of org.dvb.user.UserPreferenceChangeEvent 375

getName(String)
of org.dvb.application.AppAttributes 612

getNames()
of org.dvb.application.AppAttributes 612

getNetworkID()
of org.dvb.si.SIMonitoringEvent 430
of org.dvb.si.SINetwork 435
of org.dvb.si.SITransportStreamNIT 462

getNetworkInterface(SIDatabase)
of org.dvb.net.tuning.DvbNetworkInterfaceSIUtil 650

getNewDFC()
of org.dvb.media.DFCChangedEvent 474

getNewFormat()
of org.dvb.media.ActiveFormatDescriptionChangedEvent 468

getNewRatio()
of org.dvb.media.AspectRatioChangedEvent 469

getNewVersionNumber()
of org.dvb.dsmcc.ObjectChangeEvent 561

getNPT()
of org.dvb.dsmcc.DSMCCStream 532

getNPTRate()
of org.dvb.dsmcc.DSMCCStream 533

getNSAPAddress()
of org.dvb.dsmcc.ServiceDomain 569
of org.dvb.dsmcc.ServiceXFRReference 576

getNumerator()
of org.dvb.dsmcc.NPTRate 556

getOID()
of org.dvb.application.AppID 618

getOriginalNetworkID()
of org.dvb.si.PMTElementaryStream 391
of org.dvb.si.PMTService 392
of org.dvb.si.SIEvent 417
of org.dvb.si.SIMonitoringEvent 430
of org.dvb.si.SIService 447
of org.dvb.si.SITransportStream 458

getPassword()
of org.dvb.net.rc.ConnectionParameters 591

getPathName()
of org.dvb.dsmcc.ServiceXFRReference 576

getPcrPid()
of org.dvb.si.PMTService 392

getPermissions()
of org.dvb.io.persistent.FileAttributes 365

getPosition(String)
of org.dvb.user.Preference 373

getPositioningCapability()
of org.dvb.media.VideoPresentationControl 504

getPriority()
of org.dvb.application.AppAttributes 613
of org.dvb.io.persistent.FileAttributes 365

getProfiles()
of org.dvb.application.AppAttributes 613

getProperty(String)
of org.dvb.application.AppAttributes 613

getProperty(String, ImageObserver)
of org.dvb.ui.DVBBufferedImage 663

getProviderName()
of org.dvb.si.SIService 447

getRate()
of org.dvb.dsmcc.NPTRateChangeEvent 557

getReason()
of org.dvb.media.PresentationChangedEvent 484

getReason(int) - of org.davic.mpeg.NotAuthorizedException 231

getReceiveBufferSize(DatagramSocket)
of org.dvb.net.DatagramSocketBufferControl 584

getResult()
of org.dvb.si.SISuccessfulRetrieveEvent 454

getRGB()
of org.dvb.ui.DVBColor 670

getRGB(int, int)
of org.dvb.ui.DVBBufferedImage 663

getRGB(int, int, int, int, int[], int, int)
of org.dvb.ui.DVBBufferedImage 664

getRule()
of org.dvb.ui.DVBAlphaComposite 659

getRunningStatus()
of org.dvb.si.SIEvent 418
of org.dvb.si.SIService 447

getScaledInstance(int, int, int)
of org.dvb.ui.DVBBufferedImage 664

getScalingFactors()
of org.dvb.media.VideoTransformation 508

getService() - of org.davic.mpeg.NotAuthorizedException 231

getServiceID()
of org.dvb.si.PMTElementaryStream 391
of org.dvb.si.PMTService 392
of org.dvb.si.SIEvent 418
of org.dvb.si.SIMonitoringEvent 430
of org.dvb.si.SIService 447

getServiceLocator()
of org.dvb.application.AppAttributes 614

getServiceXFR()
of org.dvb.dsmcc.ServiceXFRErrorEvent 571
of org.dvb.dsmcc.ServiceXFRException 574

getSetupTimeEstimate()
of org.dvb.net.rc.ConnectionRCInterface 595

getShortBouquetName()
of org.dvb.si.SIBouquet 396

getShortDescription()
of org.dvb.si.SIEvent 418

getShortEventName()
of org.dvb.si.SIEvent 418

getShortNetworkName()
of org.dvb.si.SINetwork 435

getShortProviderName()
of org.dvb.si.SIService 448

getShortServiceName()
of org.dvb.si.SIService 448

getSIDatabase()
of org.dvb.si.SIDatabase 403
of org.dvb.si.SIInformation 423

getSIDatabase(NetworkInterface)
of org.dvb.net.tuning.DvbNetworkInterfaceSIUtil 650

getSigners()
of org.dvb.dsmcc.DSMCCObject 526

getSIInformationType()
of org.dvb.si.SIMonitoringEvent 430

getSIServiceLocators()
of org.dvb.si.SIBouquet 396

getSIServiceType()
of org.dvb.si.SIService 448

getSource()
of org.davic.mpeg.sections.TimeOutEvent 234
of org.dvb.dsmcc.AsynchronousLoadingEvent 521
of org.dvb.dsmcc.InvalidFormatEvent 541
of org.dvb.dsmcc.InvalidPathnameEvent 543
of org.dvb.dsmcc.LoadingAbortedEvent 545
of org.dvb.dsmcc.MPEGLDeliveryErrorEvent 546
of org.dvb.dsmcc.NotEntitledEvent 548
of org.dvb.dsmcc.NPTRateChangeEvent 557
of org.dvb.dsmcc.NPTStatusEvent 560
of org.dvb.dsmcc.ObjectChangeEvent 561
of org.dvb.dsmcc.ServerDeliveryErrorEvent 564
of org.dvb.dsmcc.ServiceXFRErrorEvent 571
of org.dvb.dsmcc.StreamEvent 578
of org.dvb.dsmcc.SuccessEvent 580
of org.dvb.event.UserEventAvailableEvent 353
of org.dvb.event.UserEventUnavailableEvent 359
of org.dvb.media.SubtitleAvailableEvent 489
of org.dvb.media.SubtitleNotAvailableEvent 491
of org.dvb.media.SubtitleNotSelectedEvent 492
of org.dvb.media.SubtitleSelectedEvent 493
of org.dvb.net.rc.RCInterfaceReleasedEvent 604
of org.dvb.net.rc.RCInterfaceReservedEvent 605
of org.dvb.si.SIMonitoringEvent 431
of org.dvb.si.SIRetrievalEvent 443
of org.dvb.ui.DVBBufferedImage 665

getStartCorner()
of org.dvb.ui.DVBTextLayoutManager 680

getStartTime()
of org.dvb.si.SIEvent 418
of org.dvb.si.SIMonitoringEvent 431

getState()
of org.dvb.application.AppProxy 620

getStream()
of org.dvb.media.CAStopEvent 473
of org.dvb.media.NoComponentSelectedEvent 482
of org.dvb.media.PresentationChangedEvent 484
of org.dvb.media.ServiceRemovedEvent 486
of org.dvb.media.StopByResourceLossEvent 488

getStreamLocator()
of org.dvb.dsmcc.DSMCCStream 533

getStreamType()
of org.dvb.si.PMTElementaryStream 391

getSubimage(int, int, int, int)
of org.dvb.ui.DVBBufferedImage 665

- getTag()
 - of org.dvb.si.Descriptor 385
- getTarget()
 - of org.dvb.net.rc.ConnectionParameters 591
- getTextualServiceIdentifiers()
 - of org.dvb.si.SIService 448
 - of org.dvb.si.TextualServiceIdentifierQuery 464
- getTextWrapping()
 - of org.dvb.ui.DVBTextLayoutManager 680
- getState()
 - of org.dvb.application.AppStateChangeEvent 635
- getTotalVideoArea()
 - of org.dvb.media.VideoPresentationControl 504
- getTotalVideoAreaOnScreen()
 - of org.dvb.media.VideoPresentationControl 504
- getTransportStreamID()
 - of org.dvb.si.PMTElementaryStream 391
 - of org.dvb.si.PMTService 393
 - of org.dvb.si.SIEvent 418
 - of org.dvb.si.SIMonitoringEvent 431
 - of org.dvb.si.SIService 449
 - of org.dvb.si.SITransportStream 458
- getType()
 - of org.dvb.application.AppAttributes 614
 - of org.dvb.event.UserEvent 350
 - of org.dvb.net.rc.RCInterface 601
 - of org.dvb.ui.DVBGraphics 674
- getType() - of org.davic.mpeg.NotAuthorizedException 231
- getUpdateTime()
 - of org.dvb.si.SIInformation 424
- getURL()
 - of org.dvb.dsmcc.DSMCCObject 527
- getURL(Locator)
 - of org.dvb.dsmcc.ServiceDomain 569
- getUserEvent()
 - of org.dvb.event.UserEventRepository 357
- getUsername()
 - of org.dvb.net.rc.ConnectionParameters 591
- getUTCTime()
 - of org.dvb.si.SITime 457
- getVersions(String)
 - of org.dvb.application.AppAttributes 614
- getVerticalAlign()
 - of org.dvb.ui.DVBTextLayoutManager 681
- getVerticalScalingFactors()
 - of org.dvb.media.VideoPresentationControl 505
- getVideoPosition()
 - of org.dvb.media.VideoTransformation 508
- getVideoSize()
 - of org.dvb.media.VideoPresentationControl 505
- getVideoTransformation()
 - of org.dvb.media.BackgroundVideoPresentationControl 470
- getVideoTransformation(int)
 - of org.dvb.media.VideoFormatControl 499
- getWidth()

of org.dvb.ui.DVBBufferedImage 665
 getWidth(ImageObserver)
 of org.dvb.ui.DVBBufferedImage 666

H

hasFailed()
 of org.dvb.application.AppStateChangeEvent 635
 hashCode()
 of org.dvb.application.AppID 618
 of org.dvb.application.AppsControlPermission 624
 hasReadApplicationAccessRight()
 of org.dvb.io.persistent.FileAccessPermissions 362
 hasReadOrganisationAccessRight()
 of org.dvb.io.persistent.FileAccessPermissions 362
 hasReadWorldAccessRight()
 of org.dvb.io.persistent.FileAccessPermissions 363
 hasValue()
 of org.dvb.user.Preference 373
 hasWriteApplicationAccessRight()
 of org.dvb.io.persistent.FileAccessPermissions 363
 hasWriteOrganisationAccessRight()
 of org.dvb.io.persistent.FileAccessPermissions 363
 hasWriteWorldAccessRight()
 of org.dvb.io.persistent.FileAccessPermissions 363
 HORIZONTAL_CENTER
 of org.dvb.ui.DVBTextLayoutManager 677
 HORIZONTAL_END_ALIGN
 of org.dvb.ui.DVBTextLayoutManager 677
 HORIZONTAL_START_ALIGN
 of org.dvb.ui.DVBTextLayoutManager 677
 HUMAN_INTERVENTION
 of org.dvb.test.DVBTest 701

I

IllegalObjectTypeException
 of org.dvb.dsmcc 538
 IllegalObjectTypeException()
 of org.dvb.dsmcc.IllegalObjectTypeException 538
 IllegalObjectTypeException(String)
 of org.dvb.dsmcc.IllegalObjectTypeException 538
 IllegalProfileParameterException
 of org.dvb.application 642
 IllegalProfileParameterException()
 of org.dvb.application.IllegalProfileParameterException 642
 IllegalProfileParameterException(String)
 of org.dvb.application.IllegalProfileParameterException 642
 implies(Permission)
 of org.dvb.application.AppsControlPermission 624
 of org.dvb.media.DripFeedPermission 478
 of org.dvb.net.ca.CAPermission 648
 of org.dvb.net.rc.RCPermission 607
 of org.dvb.net.tuning.TunerPermission 651
 IncompleteTargetException

- of org.dvb.net.rc 598
- IncompleteTargetException()
 - of org.dvb.net.rc.IncompleteTargetException 598
- IncompleteTargetException(String)
 - of org.dvb.net.rc.IncompleteTargetException 598
- init()
 - of org.dvb.application.DVBProxy 640
- InsufficientResourcesException
 - of org.dvb.dsmcc 539
- InsufficientResourcesException()
 - of org.dvb.dsmcc.InsufficientResourcesException 539
- InsufficientResourcesException(String)
 - of org.dvb.dsmcc.InsufficientResourcesException 539
- InvalidAddressException
 - of org.dvb.dsmcc 540
- InvalidAddressException()
 - of org.dvb.dsmcc.InvalidAddressException 540
- InvalidAddressException(String)
 - of org.dvb.dsmcc.InvalidAddressException 540
- InvalidFormatEvent
 - of org.dvb.dsmcc 541
- InvalidFormatEvent(DSMCCObject)
 - of org.dvb.dsmcc.InvalidFormatEvent 541
- InvalidFormatException
 - of org.dvb.dsmcc 542
- InvalidFormatException()
 - of org.dvb.dsmcc.InvalidFormatException 542
- InvalidFormatException(String)
 - of org.dvb.dsmcc.InvalidFormatException 542
- InvalidLocatorException - of org.davic.net 236
- InvalidLocatorException() - of org.davic.net.InvalidLocatorException 237
- InvalidLocatorException(String) - of org.davic.net.InvalidLocatorException 237
- InvalidPathnameEvent
 - of org.dvb.dsmcc 543
- InvalidPathnameEvent(DSMCCObject)
 - of org.dvb.dsmcc.InvalidPathnameEvent 543
- InvalidPathNameException
 - of org.dvb.dsmcc 544
- InvalidPathNameException()
 - of org.dvb.dsmcc.InvalidPathNameException 544
- InvalidPathNameException(String)
 - of org.dvb.dsmcc.InvalidPathNameException 544
- isAltDown()
 - of org.dvb.event.UserEvent 350
- isAttached()
 - of org.dvb.dsmcc.ServiceDomain 569
- isAudio()
 - of org.dvb.dsmcc.DSMCCStream 533
- isAvailableInCache()
 - of org.dvb.si.SIRequest 440
- isConnected()
 - of org.dvb.net.rc.ConnectionRCInterface 595
- isControlDown()
 - of org.dvb.event.UserEvent 350
- isData()

- of org.dvb.dsmcc.DSMCCStream 533
- isLoaded()
 - of org.dvb.dsmcc.DSMCCObject 527
- isMetaDown()
 - of org.dvb.event.UserEvent 350
- isMPEGProgram()
 - of org.dvb.dsmcc.DSMCCStream 533
- isNetworkConnectionAvailable()
 - of org.dvb.dsmcc.ServiceDomain 570
- isObjectKindKnown()
 - of org.dvb.dsmcc.DSMCCObject 527
- isOpaque()
 - of org.dvb.ui.TestOpacity 688
- isPanAndScan()
 - of org.dvb.media.VideoTransformation 508
- isPlatform()
 - of org.dvb.media.VideoFormatControl 499
- isShiftDown()
 - of org.dvb.event.UserEvent 351
- isStartable()
 - of org.dvb.application.AppAttributes 614
- isStream()
 - of org.dvb.dsmcc.DSMCCObject 527
- isStreamEvent()
 - of org.dvb.dsmcc.DSMCCObject 527
- isVideo()
 - of org.dvb.dsmcc.DSMCCStream 534
- IxcRegistry
 - of org.dvb.io.ixc 709

K

KILLED

- of org.dvb.application.DVBHTMLProxy 638

L

- LanguageNotAvailableException
 - of org.dvb.application 643
- LanguageNotAvailableException()
 - of org.dvb.application.LanguageNotAvailableException 643
- LanguageNotAvailableException(String)
 - of org.dvb.application.LanguageNotAvailableException 643
- LINE_ORIENTATION_HORIZONTAL
 - of org.dvb.ui.DVBTextLayoutManager 677
- LINE_ORIENTATION_VERTICAL
 - of org.dvb.ui.DVBTextLayoutManager 678
- LINKAGE
 - of org.dvb.si.DescriptorTag 387
- list(XletContext)
 - of org.dvb.io.ixc.IxcRegistry 710
- load()
 - of org.dvb.application.DVBProxy 640
- loadDirectoryEntry(AsynchronousLoadingEventListener)
 - of org.dvb.dsmcc.DSMCCObject 528

LOADED
 of org.dvb.application.DVBProxy 640
 LOADING
 of org.dvb.application.DVBHTMLProxy 638
 LoadingAbortedEvent
 of org.dvb.dsmcc 545
 LoadingAbortedEvent(DSMCCObject)
 of org.dvb.dsmcc.LoadingAbortedEvent 545
 LOCAL_TIME_OFFSET
 of org.dvb.si.DescriptorTag 387
 log(String, int)
 of org.dvb.test.DVBTest 702
 log(String, String)
 of org.dvb.test.DVBTest 702
 lookup(XletContext, String)
 of org.dvb.io.ixc.IxcRegistry 710

M

MHP_APPLICATION
 of org.dvb.si.SIServiceType 452
 MMIOObject - of org.davic.net.ca 243
 MOSAIC
 of org.dvb.si.DescriptorTag 387
 of org.dvb.si.SIServiceType 453
 MPEG1_AUDIO
 of org.dvb.si.PMTStreamType 394
 MPEG1_VIDEO
 of org.dvb.si.PMTStreamType 394
 MPEG2_AUDIO
 of org.dvb.si.PMTStreamType 394
 MPEG2_VIDEO
 of org.dvb.si.PMTStreamType 394
 MPEGDeliveryErrorEvent
 of org.dvb.dsmcc 546
 MPEGDeliveryErrorEvent(DSMCCObject)
 of org.dvb.dsmcc.MPEGDeliveryErrorEvent 546
 MPEGDeliveryException
 of org.dvb.dsmcc 547
 MPEGDeliveryException()
 of org.dvb.dsmcc.MPEGDeliveryException 547
 MPEGDeliveryException(String)
 of org.dvb.dsmcc.MPEGDeliveryException 547
 MULTILINGUAL_BOUQUET_NAME
 of org.dvb.si.DescriptorTag 387
 MULTILINGUAL_COMPONENT
 of org.dvb.si.DescriptorTag 387
 MULTILINGUAL_NETWORK_NAME
 of org.dvb.si.DescriptorTag 387
 MULTILINGUAL_SERVICE_NAME
 of org.dvb.si.DescriptorTag 387

N

NETWORK

- of org.dvb.si.SIMonitoringType 433
- NETWORK_NAME
 - of org.dvb.si.DescriptorTag 388
- NEW_DATABASE
 - of org.dvb.application.AppsDatabaseEvent 630
- newDatabase(AppsDatabaseEvent)
 - of org.dvb.application.AppsDatabaseEventListener 632
- newInstance(URL[])
 - of org.dvb.lang.DVBClassLoader 337
- newInstance(URL[], ClassLoader)
 - of org.dvb.lang.DVBClassLoader 337
- NoComponentSelectedEvent
 - of org.dvb.media 481
- NoComponentSelectedEvent(Controller, int, int, int, MediaLocator)
 - of org.dvb.media.NoComponentSelectedEvent 481
- NoFreeCapacityException - of org.davic.net.ca 242
- NoFreeCapacityException() - of org.davic.net.ca.NoFreeCapacityException 243
- NoFreeCapacityException(String) - of org.davic.net.ca.NoFreeCapacityException 243
- NOT_LOADED
 - of org.dvb.application.AppProxy 619
- NOT_RUNNING
 - of org.dvb.si.SIRunningStatus 445
- NotAuthorizedException - of org.davic.mpeg 230
- NotAuthorizedException() - of org.davic.mpeg.NotAuthorizedException 230
- NotAuthorizedException(String) - of org.davic.mpeg.NotAuthorizedException 230
- NotAuthorizedMediaException(ElementaryStream[], int[]) - of org.davic.media.NotAuthorizedMediaException 235
- NotAuthorizedMediaException(Service, int) - of org.davic.media.NotAuthorizedMediaException 235
- NotAuthorizedMediaException(Service, int, int)
 - of org.davic.media.NotAuthorizedMediaException 236
- NotEntitledEvent
 - of org.dvb.dsmcc 548
- NotEntitledEvent(DSMCCObject)
 - of org.dvb.dsmcc.NotEntitledEvent 548
- NotEntitledException
 - of org.dvb.dsmcc 549
- NotEntitledException()
 - of org.dvb.dsmcc.NotEntitledException 549
- NotEntitledException(String)
 - of org.dvb.dsmcc.NotEntitledException 549
- NothingToAbortException
 - of org.dvb.dsmcc 550
- NothingToAbortException()
 - of org.dvb.dsmcc.NothingToAbortException 550
- NothingToAbortException(String)
 - of org.dvb.dsmcc.NothingToAbortException 550
- notifyTextOverflow(String, HVisible, boolean, boolean)
 - of org.dvb.ui.TextOverflowListener 689
- NotLoadedException
 - of org.dvb.dsmcc 551
- NotLoadedException()
 - of org.dvb.dsmcc.NotLoadedException 551
- NotLoadedException(String)
 - of org.dvb.dsmcc.NotLoadedException 551
- NPTDiscontinuityEvent

- of org.dvb.dsmcc 552
- NPTDiscontinuityEvent(DSMCCStream, long, long)
 - of org.dvb.dsmcc.NPTDiscontinuityEvent 552
- NPTListener
 - of org.dvb.dsmcc 554
- NPTPresentEvent
 - of org.dvb.dsmcc 555
- NPTPresentEvent(DSMCCStream)
 - of org.dvb.dsmcc.NPTPresentEvent 555
- NPTRate
 - of org.dvb.dsmcc 556
- NPTRateChangeEvent
 - of org.dvb.dsmcc 557
- NPTRateChangeEvent(DSMCCStream, NPTRate)
 - of org.dvb.dsmcc.NPTRateChangeEvent 557
- NPTRemovedEvent
 - of org.dvb.dsmcc 559
- NPTRemovedEvent(DSMCCStream)
 - of org.dvb.dsmcc.NPTRemovedEvent 559
- NPTStatusEvent
 - of org.dvb.dsmcc 560
- NPTStatusEvent(DSMCCStream)
 - of org.dvb.dsmcc.NPTStatusEvent 560
- NTSC
 - of org.dvb.si.SIServiceType 453
- numberOfRemainingObjects()
 - of org.dvb.si.SIIterator 427
- NVOD_REFERENCE
 - of org.dvb.si.DescriptorTag 388
 - of org.dvb.si.SIServiceType 453
- NVOD_TIME_SHIFTED
 - of org.dvb.si.SIServiceType 453

O

- ObjectChangeEvent
 - of org.dvb.dsmcc 561
- ObjectChangeEvent(DSMCCObject, int)
 - of org.dvb.dsmcc.ObjectChangeEvent 561
- ObjectChangeListener
 - of org.dvb.dsmcc 563
- OPTION_UNSUPPORTED
 - of org.dvb.test.DVBTest 701
- org.davic.net - package 236
- org.dvb.application
 - package 609
- org.dvb.dsmcc
 - package 519
- org.dvb.event
 - package 342
- org.dvb.io.ixc
 - package 708
- org.dvb.io.persistent
 - package 361
- org.dvb.lang

- package 335
- org.dvb.media
 - package 466
- org.dvb.net
 - package 583
- org.dvb.net.ca
 - package 646
- org.dvb.net.rc
 - package 586
- org.dvb.net.tuning
 - package 649
- org.dvb.si
 - package 381
- org.dvb.test
 - package 699
- org.dvb.ui
 - package 654
- org.dvb.user
 - package 368
- OverallRepository
 - of org.dvb.event 346
- OverallRepository()
 - of org.dvb.event.OverallRepository 346
- OverallRepository(String)
 - of org.dvb.event.OverallRepository 346

P

- PAL
 - of org.dvb.si.SIServiceType 453
- PARENTAL_RATING
 - of org.dvb.si.DescriptorTag 388
- PARTIAL_TRANSPORT_STREAM
 - of org.dvb.si.DescriptorTag 388
- PASS
 - of org.dvb.test.DVBTest 701
- pause()
 - of org.dvb.application.AppProxy 620
- PAUSED
 - of org.dvb.application.AppProxy 619
- PAUSING
 - of org.dvb.si.SIRunningStatus 445
- PermissionDeniedException
 - of org.dvb.net.rc 599
- PermissionDeniedException()
 - of org.dvb.net.rc.PermissionDeniedException 599
- PermissionDeniedException(String)
 - of org.dvb.net.rc.PermissionDeniedException 599
- PMT_SERVICE
 - of org.dvb.si.SIMonitoringType 433
- PMTElementaryStream
 - of org.dvb.si 390
- PMTService
 - of org.dvb.si 392
- PMTStreamType

- of org.dvb.si 394
- POS_CAP_FULL
 - of org.dvb.media.VideoPresentationControl 502
- POS_CAP_FULL_EVEN_LINES
 - of org.dvb.media.VideoPresentationControl 502
- POS_CAP_FULL_EVEN_LINES_IF_ENTIRE_VIDEO_ON_SCREEN
 - of org.dvb.media.VideoPresentationControl 502
- POS_CAP_FULL_IF_ENTIRE_VIDEO_ON_SCREEN
 - of org.dvb.media.VideoPresentationControl 502
- POS_CAP_OTHER
 - of org.dvb.media.VideoPresentationControl 503
- postMonitoringEvent(SIMonitoringEvent)
 - of org.dvb.si.SIMonitoringListener 432
- postRetrievalEvent(SIRetrievalEvent)
 - of org.dvb.si.SIRetrievalListener 444
- Preference
 - of org.dvb.user 371
- Preference()
 - of org.dvb.user.Preference 371
- Preference(String, String)
 - of org.dvb.user.Preference 371
- Preference(String, String[])
 - of org.dvb.user.Preference 371
- prefetch()
 - of org.dvb.application.DVBHTMLProxy 638
- prefetch(DSMCCObject, String, byte)
 - of org.dvb.dsmcc.DSMCCObject 528
- prefetch(String, byte)
 - of org.dvb.dsmcc.DSMCCObject 528
- PRESENT_FOLLOWING_EVENT
 - of org.dvb.si.SIMonitoringType 433
- PresentationChangedEvent
 - of org.dvb.media 483
- PresentationChangedEvent(Controller, MediaLocator, int)
 - of org.dvb.media.PresentationChangedEvent 484
- PRIORITY_HIGH
 - of org.dvb.io.persistent.FileAttributes 364
- PRIORITY_LOW
 - of org.dvb.io.persistent.FileAttributes 364
- PRIORITY_MEDIUM
 - of org.dvb.io.persistent.FileAttributes 364
- PRIVATE_DATA_SPECIFIER
 - of org.dvb.si.DescriptorTag 388
- prompt(String, int, String)
 - of org.dvb.test.DVBTest 704

R

- RCInterface
 - of org.dvb.net.rc 600
- RCInterface()
 - of org.dvb.net.rc.RCInterface 601
- RCInterfaceManager
 - of org.dvb.net.rc 602
- RCInterfaceReleasedEvent

of org.dvb.net.rc 604
 RCInterfaceReleasedEvent(Object)
 of org.dvb.net.rc.RCInterfaceReleasedEvent 604
 RCInterfaceReservedEvent
 of org.dvb.net.rc 605
 RCInterfaceReservedEvent(Object)
 of org.dvb.net.rc.RCInterfaceReservedEvent 605
 RCPermission
 of org.dvb.net.rc 606
 RCPermission(String)
 of org.dvb.net.rc.RCPermission 606
 RCPermission(String, String)
 of org.dvb.net.rc.RCPermission 606
 read(Preference)
 of org.dvb.user.UserPreferenceManager 377
 read(Preference, Facility)
 of org.dvb.user.UserPreferenceManager 378
 rebind(XletContext, String, Remote)
 of org.dvb.io.ixc.IxcRegistry 710
 receiveEvent(AsynchronousLoadingEvent)
 of org.dvb.dsmcc.AsynchronousLoadingEventListener 522
 receiveNPTStatusEvent(NPTStatusEvent)
 of org.dvb.dsmcc.NPTListener 554
 receiveObjectChangeEvent(ObjectChangeEvent)
 of org.dvb.dsmcc.ObjectChangeListener 563
 receiveRateChangedEvent(NPTRateChangeEvent)
 of org.dvb.dsmcc.NPTListener 554
 receiveStreamEvent(StreamEvent)
 of org.dvb.dsmcc.StreamEventListener 579
 receiveUserPreferenceChangeEvent(UserPreferenceChangeEvent)
 of org.dvb.user.UserPreferenceChangeListener 376
 receiveVideoFormatEvent(VideoFormatEvent)
 of org.dvb.media.VideoFormatListener 501
 release()
 of org.dvb.net.rc.ConnectionRCInterface 595
 remove(String)
 of org.dvb.user.Preference 373
 removeAll()
 of org.dvb.user.Preference 373
 removeAllArrowKeys()
 of org.dvb.event.UserEventRepository 357
 removeAllColourKeys()
 of org.dvb.event.UserEventRepository 357
 removeAllNumericKeys()
 of org.dvb.event.UserEventRepository 357
 removeAppStateChangeListener(AppStateChangeListener)
 of org.dvb.application.AppProxy 620
 removeBouquetMonitoringListener(SIMonitoringListener, int)
 of org.dvb.si.SIDatabase 403
 removeConnectionListener(ConnectionListener)
 of org.dvb.net.rc.ConnectionRCInterface 595
 removeEventPresentFollowingMonitoringListener(SIMonitoringListener, int, int, int)
 of org.dvb.si.SIDatabase 403
 removeEventScheduleMonitoringListener(SIMonitoringListener, int, int, int)
 of org.dvb.si.SIDatabase 404

removeExclusiveAccessToAWTEvent(ResourceClient)
 of org.dvb.event.EventManager 345
 removeKey(int)
 of org.dvb.event.UserEventRepository 357
 removeListener(AppsDatabaseEventListener)
 of org.dvb.application.AppsDatabase 628
 removeNetworkMonitoringListener(SIMonitoringListener, int)
 of org.dvb.si.SIDatabase 404
 removeNPTListener(NPTListener)
 of org.dvb.dsmcc.DSMCCStream 534
 removeObjectChangeEventEventListener(ObjectChangeEventEventListener)
 of org.dvb.dsmcc.DSMCCObject 528
 removePMTServiceMonitoringListener(SIMonitoringListener, int, int, int)
 of org.dvb.si.SIDatabase 404
 removeResourceStatusEventListener(ResourceStatusListener)
 of org.dvb.event.EventManager 345
 of org.dvb.net.rc.RCInterfaceManager 603
 removeServiceMonitoringListener(SIMonitoringListener, int, int)
 of org.dvb.si.SIDatabase 405
 removeSubtitleListener(SubtitleListener)
 of org.dvb.media.SubtitlingEventControl 494
 removeTextOverflowListener(TextOverflowListener)
 of org.dvb.ui.DVBTextLayoutManager 681
 removeUserEvent(UserEvent)
 of org.dvb.event.UserEventRepository 357
 removeUserEventListener(UserEventListener)
 of org.dvb.event.EventManager 345
 removeUserPreferenceChangeListener(UserPreferenceChangeListener)
 of org.dvb.user.UserPreferenceManager 378
 removeVideoFormatListener(VideoFormatListener)
 of org.dvb.media.VideoFormatControl 499
 render(String, Graphics, HVisible, Insets)
 of org.dvb.ui.DVBTextLayoutManager 681
 RepositoryDescriptor
 of org.dvb.event 347
 reserve(ResourceClient, Object)
 of org.dvb.net.rc.ConnectionRCInterface 595
 resume()
 of org.dvb.application.AppProxy 621
 retrieveActualSINetwork(short, Object, SIRetrievalListener, short[])
 of org.dvb.si.SIDatabase 405
 retrieveActualSIServices(short, Object, SIRetrievalListener, short[])
 of org.dvb.si.SIDatabase 406
 retrieveActualSITransportStream(short, Object, SIRetrievalListener, short[])
 of org.dvb.si.SIDatabase 407
 retrieveDescriptors(short, Object, SIRetrievalListener)
 of org.dvb.si.SIBouquet 396
 of org.dvb.si.SIInformation 424
 of org.dvb.si.SINetwork 435
 retrieveDescriptors(short, Object, SIRetrievalListener, short[])
 of org.dvb.si.SIBouquet 397
 of org.dvb.si.SIInformation 424
 of org.dvb.si.SINetwork 436
 retrieveFollowingSIEvent(short, Object, SIRetrievalListener, short[])
 of org.dvb.si.SIService 449

retrievePMTElementaryStreams(short, Object, SIRetrievalListener, DvbLocator, short[])
 of org.dvb.si.SIDatabase 407
 retrievePMTElementaryStreams(short, Object, SIRetrievalListener, int, int, short[])
 of org.dvb.si.SIDatabase 408
 retrievePMTElementaryStreams(short, Object, SIRetrievalListener, short[])
 of org.dvb.si.PMTService 393
 retrievePMTService(short, Object, SIRetrievalListener, DvbLocator, short[])
 of org.dvb.si.SIDatabase 409
 retrievePMTService(short, Object, SIRetrievalListener, short[])
 of org.dvb.si.SIService 449
 retrievePMTServices(short, Object, SIRetrievalListener, int, short[])
 of org.dvb.si.SIDatabase 410
 retrievePresentSIEvent(short, Object, SIRetrievalListener, short[])
 of org.dvb.si.SIService 450
 retrieveScheduledSIEvents(short, Object, SIRetrievalListener, short[], Date, Date)
 of org.dvb.si.SIService 451
 retrieveSIBouquets(short, Object, SIRetrievalListener, int, short[])
 of org.dvb.si.SIDatabase 410
 retrieveSIBouquetTransportStreams(short, Object, SIRetrievalListener, short[])
 of org.dvb.si.SIBouquet 397
 retrieveSINetworks(short, Object, SIRetrievalListener, int, short[])
 of org.dvb.si.SIDatabase 411
 retrieveSIService(short, Object, SIRetrievalListener, DvbLocator, short[])
 of org.dvb.si.SIDatabase 412
 retrieveSIService(short, Object, SIRetrievalListener, short[])
 of org.dvb.si.SIEvent 419
 retrieveSIServices(short, Object, SIRetrievalListener, int, int, int, short[])
 of org.dvb.si.SIDatabase 413
 retrieveSIServices(short, Object, SIRetrievalListener, short[])
 of org.dvb.si.SITransportStream 459
 retrieveSITimeFromTDT(short, Object, SIRetrievalListener)
 of org.dvb.si.SIDatabase 413
 retrieveSITimeFromTOT(short, Object, SIRetrievalListener, short[])
 of org.dvb.si.SIDatabase 414
 retrieveSITransportStreamDescription(short, Object, SIRetrievalListener, short[])
 of org.dvb.si.SIDatabase 415
 retrieveSITransportStreams(short, Object, SIRetrievalListener, short[])
 of org.dvb.si.SINetwork 436
 RUNNING
 of org.dvb.si.SIRunningStatus 445
 RunningApplicationsFilter
 of org.dvb.application 644
 RunningApplicationsFilter()
 of org.dvb.application.RunningApplicationsFilter 644

S

SATELLITE_DELIVERY_SYSTEM
 of org.dvb.si.DescriptorTag 388
 SCHEDULED_EVENT
 of org.dvb.si.SIMonitoringType 433
 SECAM
 of org.dvb.si.SIServiceType 453
 selectServiceMediaComponents(Locator)
 of org.dvb.media.DVBMediaSelectControl 480

ServerDeliveryErrorEvent
 of org.dvb.dsmcc 564
 ServerDeliveryErrorEvent(DSMCCObject)
 of org.dvb.dsmcc.ServerDeliveryErrorEvent 564
 ServerDeliveryException
 of org.dvb.dsmcc 565
 ServerDeliveryException()
 of org.dvb.dsmcc.ServerDeliveryException 565
 ServerDeliveryException(String)
 of org.dvb.dsmcc.ServerDeliveryException 565
 SERVICE
 of org.dvb.si.DescriptorTag 388
 of org.dvb.si.SIMonitoringType 433
 SERVICE_LIST
 of org.dvb.si.DescriptorTag 388
 SERVICE_MOVE
 of org.dvb.si.DescriptorTag 388
 ServiceDomain
 of org.dvb.dsmcc 566
 ServiceDomain()
 of org.dvb.dsmcc.ServiceDomain 566
 ServiceRemovedEvent
 of org.dvb.media 485
 ServiceRemovedEvent(Controller)
 of org.dvb.media.ServiceRemovedEvent 485
 ServiceRemovedEvent(Controller, int, int, int, MediaLocator)
 of org.dvb.media.ServiceRemovedEvent 485
 ServiceXFRErrorEvent
 of org.dvb.dsmcc 571
 ServiceXFRErrorEvent(DSMCCObject, ServiceXFRReference)
 of org.dvb.dsmcc.ServiceXFRErrorEvent 571
 ServiceXFRException
 of org.dvb.dsmcc 573
 ServiceXFRException(byte[], String)
 of org.dvb.dsmcc.ServiceXFRException 573
 ServiceXFRException(Locator, int, String)
 of org.dvb.dsmcc.ServiceXFRException 573
 ServiceXFRReference
 of org.dvb.dsmcc 575
 ServiceXFRReference(byte[], String)
 of org.dvb.dsmcc.ServiceXFRReference 575
 ServiceXFRReference(Locator, int, String)
 of org.dvb.dsmcc.ServiceXFRReference 575
 setClipRegion(Rectangle)
 of org.dvb.media.VideoPresentationControl 505
 of org.dvb.media.VideoTransformation 508
 setColor(Color)
 of org.dvb.ui.DVBGraphics 674
 setDVBComposite(DVBAlphaComposite)
 of org.dvb.ui.DVBGraphics 674
 setExpirationDate(Date)
 of org.dvb.io.persistent.FileAttributes 365
 setFileAttributes(FileAttributes, File)
 of org.dvb.io.persistent.FileAttributes 365
 setHorizontalAlign(int)

- of org.dvb.ui.DVBTextLayoutManager 681
- setHorizontalTabSpacing(int)
 - of org.dvb.ui.DVBTextLayoutManager 682
- setInsets(Insets)
 - of org.dvb.ui.DVBTextLayoutManager 682
- setLetterSpace(int)
 - of org.dvb.ui.DVBTextLayoutManager 682
- setLineOrientation(int)
 - of org.dvb.ui.DVBTextLayoutManager 682
- setLineSpace(int)
 - of org.dvb.ui.DVBTextLayoutManager 682
- setMostFavourite(String)
 - of org.dvb.user.Preference 373
- setPermissions(boolean, boolean, boolean, boolean, boolean, boolean)
 - of org.dvb.io.persistent.FileAccessPermissions 363
- setPermissions(FileAccessPermissions)
 - of org.dvb.io.persistent.FileAttributes 366
- setPriority(int)
 - of org.dvb.io.persistent.FileAttributes 366
- setReceiveBufferSize(DatagramSocket, int)
 - of org.dvb.net.DatagramSocketBufferControl 584
- setRetrievalMode(int)
 - of org.dvb.dsmcc.DSMCCObject 529
- setRGB(int, int, int)
 - of org.dvb.ui.DVBBufferedImage 666
- setRGB(int, int, int, int, int[], int, int)
 - of org.dvb.ui.DVBBufferedImage 666
- setScalingFactors(float, float)
 - of org.dvb.media.VideoTransformation 508
- setStartCorner(int)
 - of org.dvb.ui.DVBTextLayoutManager 682
- setTarget(ConnectionParameters)
 - of org.dvb.net.rc.ConnectionRCInterface 596
- setTargetToDefault()
 - of org.dvb.net.rc.ConnectionRCInterface 596
- setTextWrapping(boolean)
 - of org.dvb.ui.DVBTextLayoutManager 683
- setVerticalAlign(int)
 - of org.dvb.ui.DVBTextLayoutManager 683
- setVideoPosition(HScreenPoint)
 - of org.dvb.media.VideoTransformation 509
- setVideoTransformation(VideoTransformation)
 - of org.dvb.media.BackgroundVideoPresentationControl 470
- SHORT_EVENT
 - of org.dvb.si.DescriptorTag 389
- SHORT_SMOOTHING_BUFFER
 - of org.dvb.si.DescriptorTag 389
- SIBouquet
 - of org.dvb.si 395
- SIDatabase
 - of org.dvb.si 399
- SIDatabase()
 - of org.dvb.si.SIDatabase 399
- SIEvent
 - of org.dvb.si 416

SException
of org.dvb.si 420

SException()
of org.dvb.si.SException 420

SException(String)
of org.dvb.si.SException 420

SIllegalArgumentException
of org.dvb.si 421

SIllegalArgumentException()
of org.dvb.si.SIllegalArgumentException 421

SIllegalArgumentException(String)
of org.dvb.si.SIllegalArgumentException 421

SInformation
of org.dvb.si 422

SInvalidPeriodException
of org.dvb.si 426

SInvalidPeriodException()
of org.dvb.si.SInvalidPeriodException 426

SInvalidPeriodException(String)
of org.dvb.si.SInvalidPeriodException 426

SIterator
of org.dvb.si 427

SILackOfResourcesEvent
of org.dvb.si 428

SILackOfResourcesEvent(Object, SRequest)
of org.dvb.si.SILackOfResourcesEvent 428

SIMonitoringEvent
of org.dvb.si 429

SIMonitoringEvent(SIDatabase, byte, int, int, int, int, Date, Date)
of org.dvb.si.SIMonitoringEvent 429

SIMonitoringListener
of org.dvb.si 432

SIMonitoringType
of org.dvb.si 433

SINetwork
of org.dvb.si 434

SINotInCacheEvent
of org.dvb.si 438

SINotInCacheEvent(Object, SRequest)
of org.dvb.si.SINotInCacheEvent 438

SIOBJECTNotInTableEvent
of org.dvb.si 439

SIOBJECTNotInTableEvent(Object, SRequest)
of org.dvb.si.SIOBJECTNotInTableEvent 439

SRequest
of org.dvb.si 440

SRequest()
of org.dvb.si.SRequest 440

SRequestCancelledEvent
of org.dvb.si 441

SRequestCancelledEvent(Object, SRequest)
of org.dvb.si.SRequestCancelledEvent 441

SRetrievalEvent
of org.dvb.si 442

SRetrievalEvent(Object, SRequest)

of org.dvb.si.SIRetrievalEvent 442
 SIRetrievalListener
 of org.dvb.si 444
 SIRunningStatus
 of org.dvb.si 445
 SIService
 of org.dvb.si 446
 SIServiceType
 of org.dvb.si 452
 SISuccessfulRetrieveEvent
 of org.dvb.si 454
 SISuccessfulRetrieveEvent(Object, SIRequest, SIIterator)
 of org.dvb.si.SISuccessfulRetrieveEvent 454
 SITableNotFoundEvent
 of org.dvb.si 455
 SITableNotFoundEvent(Object, SIRequest)
 of org.dvb.si.SITableNotFoundEvent 455
 SITableUpdatedEvent
 of org.dvb.si 456
 SITableUpdatedEvent(Object, SIRequest)
 of org.dvb.si.SITableUpdatedEvent 456
 SITime
 of org.dvb.si 457
 SITransportStream
 of org.dvb.si 458
 SITransportStreamBAT
 of org.dvb.si 460
 SITransportStreamDescription
 of org.dvb.si 461
 SITransportStreamNIT
 of org.dvb.si 462
 SIUtil
 of org.dvb.si 463
 size()
 of org.dvb.application.AppsDatabase 628
 SRC
 of org.dvb.ui.DVBAlphaComposite 657
 Src
 of org.dvb.ui.DVBAlphaComposite 657
 SRC_IN
 of org.dvb.ui.DVBAlphaComposite 657
 SRC_OUT
 of org.dvb.ui.DVBAlphaComposite 657
 SRC_OVER
 of org.dvb.ui.DVBAlphaComposite 658
 SrcIn
 of org.dvb.ui.DVBAlphaComposite 658
 SrcOut
 of org.dvb.ui.DVBAlphaComposite 658
 SrcOver
 of org.dvb.ui.DVBAlphaComposite 658
 start()
 of org.dvb.application.AppProxy 621
 of org.dvb.media.DripFeedDataSource 477
 start(String[])

of org.dvb.application.AppProxy 621
 START_CORNER_LOWER_LEFT
 of org.dvb.ui.DVBTextLayoutManager 678
 START_CORNER_LOWER_RIGHT
 of org.dvb.ui.DVBTextLayoutManager 678
 START_CORNER_UPPER_LEFT
 of org.dvb.ui.DVBTextLayoutManager 678
 START_CORNER_UPPER_RIGHT
 of org.dvb.ui.DVBTextLayoutManager 678
 STARTED
 of org.dvb.application.AppProxy 620
 STARTS_IN_A_FEW_SECONDS
 of org.dvb.si.SIRunningStatus 445
 startTrigger(Date)
 of org.dvb.application.DVBHTMLProxy 639
 stateChange(AppStateChangeEvent)
 of org.dvb.application.AppStateChangeListener 636
 stop()
 of org.dvb.media.DripFeedDataSource 477
 stop(boolean)
 of org.dvb.application.AppProxy 622
 StopByResourceLossEvent
 of org.dvb.media 487
 StopByResourceLossEvent(Controller, int, int, int, MediaLocator)
 of org.dvb.media.StopByResourceLossEvent 487
 STREAM_IDENTIFIER
 of org.dvb.si.DescriptorTag 389
 STREAM_UNAVAILABLE
 of org.dvb.media.PresentationChangedEvent 483
 StreamEvent
 of org.dvb.dsmcc 577
 StreamEvent(DSMCCStreamEvent, long, String, int, byte[])
 of org.dvb.dsmcc.StreamEvent 577
 StreamEventListener
 of org.dvb.dsmcc 579
 STUFFING
 of org.dvb.si.DescriptorTag 389
 subscribe(String, StreamEventListener)
 of org.dvb.dsmcc.DSMCCStreamEvent 536
 SubtitleAvailableEvent
 of org.dvb.media 489
 SubtitleAvailableEvent(SubtitlingLanguageControl)
 of org.dvb.media.SubtitleAvailableEvent 489
 SubtitleListener
 of org.dvb.media 490
 SubtitleNotAvailableEvent
 of org.dvb.media 491
 SubtitleNotAvailableEvent(SubtitlingLanguageControl)
 of org.dvb.media.SubtitleNotAvailableEvent 491
 SubtitleNotSelectedEvent
 of org.dvb.media 492
 SubtitleNotSelectedEvent(SubtitlingLanguageControl)
 of org.dvb.media.SubtitleNotSelectedEvent 492
 SubtitleSelectedEvent
 of org.dvb.media 493

SubtitleSelectedEvent(SubtitleLanguageControl)
 of org.dvb.media.SubtitleSelectedEvent 493
 subtitleStatusChanged(EventObject)
 of org.dvb.media.SubtitleListener 490
 SUBTITLING
 of org.dvb.si.DescriptorTag 389
 SubtitlingEventControl
 of org.dvb.media 494
 SuccessEvent
 of org.dvb.dsmcc 580
 SuccessEvent(DSMCCObject)
 of org.dvb.dsmcc.SuccessEvent 580
 supportsArbitraryHorizontalScaling()
 of org.dvb.media.VideoPresentationControl 505
 supportsArbitraryVerticalScaling()
 of org.dvb.media.VideoPresentationControl 506
 supportsClipping()
 of org.dvb.media.VideoPresentationControl 506
 synchronousLoad()
 of org.dvb.dsmcc.DSMCCObject 529

T

TELEPHONE
 of org.dvb.si.DescriptorTag 389
 TELETEXT
 of org.dvb.si.DescriptorTag 389
 of org.dvb.si.SIServiceType 453
 terminate(String, int)
 of org.dvb.test.DVBTest 705
 TERRESTRIAL_DELIVERY_SYSTEM
 of org.dvb.si.DescriptorTag 389
 TestOpacity
 of org.dvb.ui 688
 TextOverflowListener
 of org.dvb.ui 689
 TextualServiceIdentifierQuery
 of org.dvb.si 464
 TIME_SHIFTED_EVENT
 of org.dvb.si.DescriptorTag 389
 TIME_SHIFTED_SERVICE
 of org.dvb.si.DescriptorTag 389
 TimeOutEvent
 of org.davic.mpeg.sections 234
 TimeOutEvent(SectionFilter, Object)
 of org.davic.mpeg.sections.TimeOutEvent 234
 toString()
 of org.dvb.application.AppID 618
 of org.dvb.ui.DVBBufferedImage 667
 of org.dvb.ui.DVBColor 671
 of org.dvb.ui.DVBGraphics 675
 of org.dvb.user.Preference 373
 trigger(Date, Object)
 of org.dvb.application.DVBHTMLProxy 639
 TunerPermission

of org.dvb.net.tuning 651
 TunerPermission(String)
 of org.dvb.net.tuning.TunerPermission 651
 TunerPermission(String, String)
 of org.dvb.net.tuning.TunerPermission 651
 TYPE_ADVANCED
 of org.dvb.ui.DVBBufferedImage 660
 TYPE_BASE
 of org.dvb.ui.DVBBufferedImage 660
 TYPE_CATV
 of org.dvb.net.rc.RCInterface 600
 TYPE_DECT
 of org.dvb.net.rc.RCInterface 600
 TYPE_ISDN
 of org.dvb.net.rc.RCInterface 600
 TYPE_LMDS
 of org.dvb.net.rc.RCInterface 600
 TYPE_MATV
 of org.dvb.net.rc.RCInterface 600
 TYPE_PSTN
 of org.dvb.net.rc.RCInterface 601
 TYPE_RCS
 of org.dvb.net.rc.RCInterface 601

U

UEF_KEY_EVENT
 of org.dvb.event.UserEvent 348
 unbind(XletContext, String)
 of org.dvb.io.ixc.IxcRegistry 711
 UNDEFINED
 of org.dvb.si.SIRunningStatus 445
 UNKNOWN
 of org.dvb.si.SIServiceType 453
 UnknownEventException
 of org.dvb.dsmcc 581
 UnknownEventException()
 of org.dvb.dsmcc.UnknownEventException 581
 UnknownEventException(String)
 of org.dvb.dsmcc.UnknownEventException 581
 unload()
 of org.dvb.dsmcc.DSMCCObject 529
 UNRESOLVED
 of org.dvb.test.DVBTest 702
 unsubscribe(int, StreamEventListener)
 of org.dvb.dsmcc.DSMCCStreamEvent 536
 unsubscribe(String, StreamEventListener)
 of org.dvb.dsmcc.DSMCCStreamEvent 537
 UnsupportedDrawingOperationException
 of org.dvb.ui 690
 UnsupportedDrawingOperationException(String)
 of org.dvb.ui.UnsupportedDrawingOperationException 690
 UnsupportedPreferenceException
 of org.dvb.user 374
 UnsupportedPreferenceException()

- of org.dvb.user.UnsupportedPreferenceException 374
- UnsupportedPreferenceException(String)
 - of org.dvb.user.UnsupportedPreferenceException 374
- UNTESTED
 - of org.dvb.test.DVBTest 702
- UserEvent
 - of org.dvb.event 348
- UserEvent(Object, int, char)
 - of org.dvb.event.UserEvent 348
- UserEvent(Object, int, int, int, int)
 - of org.dvb.event.UserEvent 349
- UserEventAvailableEvent
 - of org.dvb.event 352
- UserEventAvailableEvent(Object)
 - of org.dvb.event.UserEventAvailableEvent 352
- UserEventListener
 - of org.dvb.event 354
- userEventReceived(UserEvent)
 - of org.dvb.event.UserEventListener 354
- UserEventRepository
 - of org.dvb.event 355
- UserEventRepository(String)
 - of org.dvb.event.UserEventRepository 355
- UserEventUnavailableEvent
 - of org.dvb.event 358
- UserEventUnavailableEvent(Object)
 - of org.dvb.event.UserEventUnavailableEvent 358
- UserPreferenceChangeEvent
 - of org.dvb.user 375
- UserPreferenceChangeEvent(String)
 - of org.dvb.user.UserPreferenceChangeEvent 375
- UserPreferenceChangeListener
 - of org.dvb.user 376
- UserPreferenceManager
 - of org.dvb.user 377
- UserPreferencePermission
 - of org.dvb.user 379
- UserPreferencePermission(String)
 - of org.dvb.user.UserPreferencePermission 379
- UserPreferencePermission(String, String)
 - of org.dvb.user.UserPreferencePermission 379

V

- VERTICAL_CENTER
 - of org.dvb.ui.DVBTextLayoutManager 678
- VERTICAL_END_ALIGN
 - of org.dvb.ui.DVBTextLayoutManager 678
- VERTICAL_START_ALIGN
 - of org.dvb.ui.DVBTextLayoutManager 678
- VideoFormatControl
 - of org.dvb.media 495
- VideoFormatEvent
 - of org.dvb.media 500
- VideoFormatEvent(Object)

of org.dvb.media.VideoFormatEvent 500

VideoFormatListener

of org.dvb.media 501

VideoPresentationControl

of org.dvb.media 502

VideoTransformation

of org.dvb.media 507

VideoTransformation()

of org.dvb.media.VideoTransformation 507

VideoTransformation(Rectangle, float, float, HScreenPoint)

of org.dvb.media.VideoTransformation 507

W

write(Preference)

of org.dvb.user.UserPreferenceManager 378

History

Document history		
V1.1.1	July 2000	Published (Withdrawn)
V1.1.2	October 2001	Published (Withdrawn)
V1.2.1	June 2002	Published

ETSI IPR Policy

Extracted from the ETSI Rules of Procedure, 22 November 2000

ANNEX 6: ETSI INTELLECTUAL PROPERTY RIGHTS POLICY

1 Introduction

The General Assembly of ETSI has established the following Intellectual Property Rights POLICY.

2 Definitions

Terms in the POLICY that are written in capital letters shall have the meaning set forth in clause 15 entitled DEFINITIONS.

3 Policy Objectives

3.1 STANDARDS and TECHNICAL SPECIFICATIONS shall be based on solutions which best meet the technical objectives of the European telecommunications sector, as defined by the General Assembly. In order to further this objective the ETSI IPR POLICY seeks to reduce the risk to ETSI, MEMBERS, and others applying ETSI STANDARDS and TECHNICAL SPECIFICATIONS, that investment in the preparation, adoption and application of STANDARDS could be wasted as a result of an ESSENTIAL IPR for a STANDARD or TECHNICAL SPECIFICATION being unavailable. In achieving this objective, the ETSI IPR POLICY seeks a balance between the needs of standardization for public use in the field of telecommunications and the rights of the owners of IPRs

3.2 IPR holders whether members of ETSI and their AFFILIATES or third parties, should be adequately and fairly rewarded for the use of their IPRs in the implementation of STANDARDS and TECHNICAL SPECIFICATIONS.

3.3 ETSI shall take reasonable measures to ensure, as far as possible, that its activities which relate to the preparation, adoption and application of STANDARDS and TECHNICAL SPECIFICATIONS, enable STANDARDS and TECHNICAL SPECIFICATIONS to be available to potential users in accordance with the general principles of standardization.

TOP

4 Disclosure of IPRs

4.1 Each MEMBER shall use its reasonable endeavours to timely inform ETSI of ESSENTIAL IPRs it becomes aware of. In particular, a MEMBER submitting a technical proposal for a STANDARD or TECHNICAL SPECIFICATION shall, on a bona fide basis, draw the attention of ETSI to any of that MEMBER's IPR which might be ESSENTIAL if that proposal is adopted.

4.2 The obligations pursuant to Clause 4.1 above do however not imply any obligation on MEMBERS to conduct IPR searches.

5 Procedures for Committees

ETSI shall establish guidelines for the chairmen of COMMITTEES with respect to ESSENTIAL IPRs.

6 Availability of Licences

6.1 When an ESSENTIAL IPR relating to a particular STANDARD or TECHNICAL SPECIFICATION is brought to the attention of ETSI, the Director-General of ETSI shall immediately request the owner to give within three months an undertaking in writing that it is prepared to grant irrevocable licences on fair, reasonable and non-discriminatory terms and conditions under such IPR to at least the following extent:

- MANUFACTURE, including the right to make or have made customized components and sub-systems to the licensee's own

design for use in MANUFACTURE;

- sell, lease, or otherwise dispose of EQUIPMENT so MANUFACTURED;
- repair, use, or operate EQUIPMENT; and
- use METHODS.

The above undertaking may be made subject to the condition that those who seek licences agree to reciprocate.

6.2 At the request of the European Commission and/or EFTA, initially for a specific STANDARD or TECHNICAL SPECIFICATION or a class of STANDARDS/TECHNICAL SPECIFICATIONS, ETSI shall arrange to have carried out in a competent and timely manner an investigation including an IPR search, with the objective of ascertaining whether IPRs exist or are likely to exist which may be or may become ESSENTIAL to a proposed STANDARD or TECHNICAL SPECIFICATIONS and the possible terms and conditions of licences for such IPRs. This shall be subject to the European Commission and/or EFTA meeting all reasonable expenses of such an investigation, in accordance with detailed arrangements to be worked out with the European Commission and/or EFTA prior to the investigation being undertaken.

TOP

7 Information on IPR by ETSI

7.1 Any published STANDARD or TECHNICAL SPECIFICATION shall include information pertaining to ESSENTIAL IPRs which are brought to the attention of ETSI prior to such publication.

7.2 ETSI shall establish appropriate procedures to allow access to information at any time with respect to ESSENTIAL IPRs which have been brought to the attention of ETSI.

8 Non-availability of Licences

8.1 MEMBERS' refusal to license

8.1.1 Where a MEMBER notifies ETSI that it is not prepared to license an IPR in respect of a STANDARD or TECHNICAL SPECIFICATION, the General Assembly shall review the requirement for that STANDARD or TECHNICAL SPECIFICATION and satisfy itself that a viable alternative technology is available for the STANDARD or TECHNICAL SPECIFICATION which:

- is not blocked by that IPR; and
- satisfies ETSI's requirements.

8.1.2 Where, in the opinion of the General Assembly, no such viable alternative technology exists, work on the STANDARD or TECHNICAL SPECIFICATION shall cease, and the Director-General of ETSI shall request that MEMBER to reconsider its position. If the MEMBER decides not to withdraw its refusal to license the IPR, it shall inform the Director-General of ETSI of its decision and provide a written explanation of its reasons for refusing to license that IPR, within three months of its receipt of the Director-General's request.

The Director-General shall then send the MEMBER's explanation together with relevant extracts from the minutes of the General Assembly to the ETSI Counsellors for their consideration.

8.2 Non-availability of licences from third parties

Where, in respect of a STANDARD or TECHNICAL SPECIFICATION, ETSI becomes aware that licences are not available from a third party in accordance with Clause 6.1 above, that STANDARD or TECHNICAL SPECIFICATION shall be referred to the Director-General of ETSI for further consideration in accordance with the following procedure:

- i) The Director-General shall request full supporting details from any MEMBER who has complained that licences are not available in accordance with Clause 6.1 above.
- ii) The Director-General shall write to the IPR owner concerned for an explanation and request that licences be granted according to Clause 6.1 above.
- iii) Where the IPR owner refuses the Director-General's request or does not answer the letter within three months, the

Director-General shall inform the General Assembly. A vote shall be taken in the General Assembly on an individual weighted basis to immediately refer the STANDARD or TECHNICAL SPECIFICATION to the relevant COMMITTEE to modify it so that the IPR is no longer ESSENTIAL.

iv) Where the vote in the General Assembly does not succeed, then the General Assembly shall, where appropriate, consult the ETSI Counsellors with a view to finding a solution to the problem. In parallel, the General Assembly may request appropriate MEMBERS to use their good offices to find a solution to the problem.

v) Where (iv) does not lead to a solution, then the General Assembly shall request the European Commission to see what further action may be appropriate, including non-recognition of the STANDARD or TECHNICAL SPECIFICATION in question.

In carrying out the foregoing procedure due account shall be taken of the interest of the enterprises that have invested in the implementation of the STANDARD or TECHNICAL SPECIFICATION in question.

TOP

9 ETSI ownership of IPRs

9.1 The ownership of the copyright in STANDARDS and TECHNICAL SPECIFICATIONS documentation and reports created by ETSI or any of its COMMITTEES shall vest in ETSI but due acknowledgement shall be given to copyrights owned by third parties that are identifiable in ETSI copyrighted works.

9.2 In respect of IPRs other than copyright in STANDARDS and TECHNICAL SPECIFICATIONS documentation and reports, ETSI shall only seek ownership of IPRs generated either by its employees or by secondees to ETSI from organizations who are not MEMBERS.

9.3 ETSI shall, on request by a non-member, grant licences to that non-member on fair and reasonable terms and conditions in respect of any IPRs, other than those referred to in Clause 9.1 above, owned by ETSI. MEMBERS shall be allowed to use IPRs owned by ETSI free of charge.

10 Confidentiality

The proceedings of a COMMITTEE shall be regarded as non confidential except as expressly provided below and all information submitted to a COMMITTEE shall be treated as if non-confidential and shall be available for public inspection unless:

- the information is in written or other tangible form; and
- the information is identified in writing, when submitted, as confidential; and
- the information is first submitted to, and accepted by, the chairman of the COMMITTEE as confidential.

CONFIDENTIAL INFORMATION incorporated in a STANDARD or TECHNICAL SPECIFICATION shall be regarded as non-confidential by ETSI and its MEMBERS, from the date on which the STANDARD or TECHNICAL SPECIFICATION is published.

11 Reproduction of Standards Documentation

MEMBERS may make copies of STANDARDS and TECHNICAL SPECIFICATIONS documentation produced by ETSI for their own use free of charge but may not distribute such copies to others.

TOP

12 Law and Regulation

The POLICY shall be governed by the laws of France. However, no MEMBER shall be obliged by the POLICY to commit a breach of the laws or regulations of its country or to act against supranational laws or regulations applicable to its country insofar as

derogation by agreement between parties is not permitted by such laws.

Any right granted to, and any obligation imposed on, a MEMBER which derives from French law and which are not already contained in the national or supranational law applicable to that MEMBER is to be understood as being of solely a contractual nature.

13 Policy Decisions

Without prejudice to ETSI's Statutes and Rules of Procedure, no decisions shall be taken by ETSI in relation to implementation of the POLICY unless supported by a 71 % majority of the weighted individual votes cast by MEMBERS.

14 Violation of Policy

Any violation of the POLICY by a MEMBER shall be deemed to be a breach, by that MEMBER, of its obligations to ETSI. The ETSI General Assembly shall have the authority to decide the action to be taken, if any, against the MEMBER in breach, in accordance with the ETSI Statutes.

15 Definitions

1 "AFFILIATE" of a first legal entity means any other legal entity:

- directly or indirectly owning or controlling the first legal entity, or
- under the same direct or indirect ownership or control as the first legal entity, or
- directly or indirectly owned or controlled by the first legal entity,

for so long as such ownership or control lasts.

Ownership or control shall exist through the direct or indirect:

- ownership of more than 50 % of the nominal value of the issued equity share capital or of more than 50 % of the shares entitling the holders to vote for the election of directors or persons performing similar functions, or
- right by any other means to elect or appoint directors, or persons who collectively can exercise such control. A state, a division of a state or other public entity operating under public law, or any legal entity, linked to the first legal entity solely through a state or any division of a state or other public entity operating under public law, shall be deemed to fall outside the definition of an AFFILIATE.

2 "COMMITTEE" shall mean any Technical Body of ETSI and shall include ETSI Projects, Technical Committees, ETSI Partnership Projects, and their Working Groups.

3 "CONFIDENTIAL INFORMATION" shall mean all information deemed to be confidential pursuant to Clause 10 of the POLICY disclosed directly or indirectly to the MEMBER.

4 "EQUIPMENT" shall mean any system, or device fully conforming to a STANDARD.

5 "METHODS" shall mean any method or operation fully conforming to a STANDARD.

6 "ESSENTIAL" as applied to IPR means that it is not possible on technical (but not commercial) grounds, taking into account normal technical practice and the state of the art generally available at the time of standardization, to make, sell, lease, otherwise dispose of, repair, use or operate EQUIPMENT or METHODS which comply with a STANDARD without infringing that IPR. For the avoidance of doubt in exceptional cases where a STANDARD can only be implemented by technical solutions, all of which are infringements of IPRs, all such IPRs shall be considered ESSENTIAL.

7 "IPR" shall mean any intellectual property right conferred by statute law including applications therefor other than trademarks. For the avoidance of doubt rights relating to get up, confidential information, trade secrets or the like are excluded from the definition of IPR.

8 "MANUFACTURE", shall mean production of EQUIPMENT.

9 "MEMBER" shall mean a member or associate member of ETSI. References to a MEMBER shall wherever the context permits be interpreted as references to that MEMBER and its AFFILIATES.

10 "POLICY" shall mean ETSI's Intellectual Property Rights Policy.

11 "STANDARD" shall mean any standard adopted by ETSI including options therein or amended versions and shall include European Standards (ENs) (telecommunications series), ETSI Standards (ESs), Common Technical Regulations (CTRs) which are taken from ENs (telecommunications series) and including drafts of any of the foregoing, and documents made under the previous nomenclature, including ETSS, I-ETSS, parts of NETs and TBRs, the technical specifications of which are available to all MEMBERS, but not including any standards, or parts thereof, not made by ETSI.

The date on which a STANDARD is considered to be adopted by ETSI for the purposes of this POLICY shall be the date on which the technical content of that STANDARD was available to all MEMBERS.

12 "TECHNICAL SPECIFICATION" shall mean any Technical Specification (TS) adopted by ETSI including options therein or amended version including drafts, the Technical Specifications of which are available to all MEMBERS, but not including any technical specifications, or parts thereof, not made by ETSI.

The date on which a TECHNICAL SPECIFICATION is considered to be adopted by ETSI for the purposes of this POLICY shall be the date on which the technical content of that TECHNICAL SPECIFICATION was available to all MEMBERS.

