ETSI TS 101 812 V1.1.1 (2000-07)

Technical Specification

Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.0

Reference
DTS/JTC-DVB-112

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

Contents

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 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/ipr).

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 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.

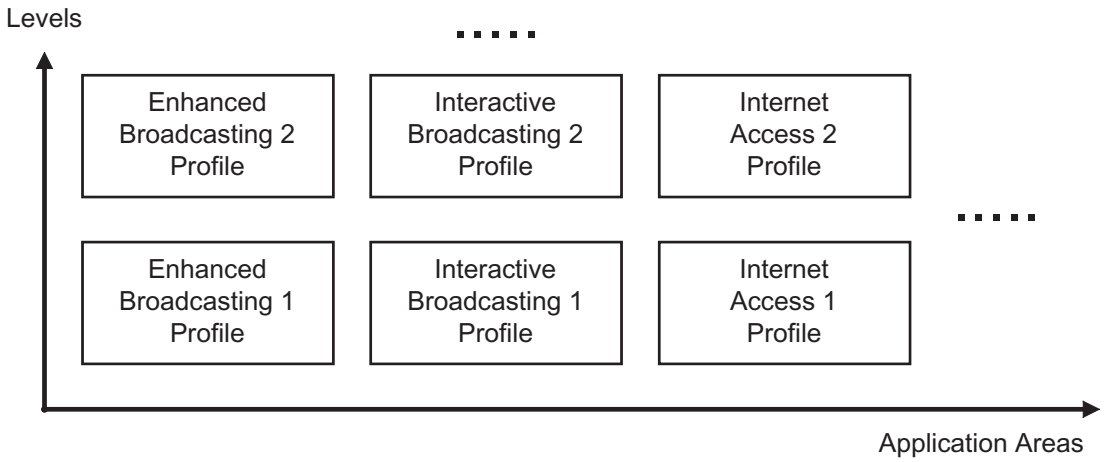


Figure 0 : Application areas and levels of profiles

Fig. 0 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 1is 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 [A1]. It relies on the use of appropriate DVB specifications for digital video broadcast and associated interactive services TR 101 200 [49]. 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, edition number, version number, etc.) 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): Errata in external references" on page 167. 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.

	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]	ETR 154	3	DVB Implementation Guidelines for the use of MPEG-2 Systems, Video and Audio in Satellite, Cable and Terrestrial Broadcasting Applications.	[6]
[10]	ETR 162		Digital broadcasting systems for television, sound and data services; Allocation of Service Information (SI) codes for Digital Video Broadcasting (DVB) systems	[6]
[11]	ETR 211	2	Digital Video Broadcasting (DVB); Guidelines on implementation and usage of Service Information (SI)	[6]
[12]	EN 300 472	1.2.2	DVB Specification for conveying ITU-R System B Teletext in DVB bitstreams.	[6]
[13]	EN 300 743	1	Digital Video Broadcasting (DVB), DVB subtitling	[6]
[14]	ETS 300 800	1	DVB Interaction Channel for Cable TV distribution systems	[6]
[15]	ETS 300 801	1	DVB Interaction Channel through PSTN/ISDN	[6]
[16]	ETS 300 802	1	Network Independent Protocols for Interactive Services	[6]
[17]	GIF 89a		GRAPHICS INTERCHANGE FORMAT(sm) Version 89a, (c)1987,1988,1989,1990 Copyright CompuServe Incorporated Columbus, Ohio 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" (also known as Unicode, UTF)	[1]
[19]	ISO 639.2		Code for the representation of names of languages	[1]
[20]	ISO 8859		Information processing - 8-bit single-byte coded graphic character sets, Latin alphabets	[1]
[21]	ISO/IEC 10918-1		Digital compression encoding of continuous-tone still images (JPEG). 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]

	Reference	Edition	Description	Note
[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	2nd 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]	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.	[2]
[32]	Java Class Libraries Vol. 2	ISBN 0-201-31003-1	The Java Class Libraries, Second Edition, Volume 2 by Patrick Chan and Rosanna Lee. See A, "(normative): Errata in external references" on page 167.	[2]
[33]	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]
[34]	Java Media Player Specification.	1.0, Sept 2, 1997	Sun Microsystems Java Media Player Specification. (javadoc). http://java.sun.com/products/java-media/jmf/forDevelopers/playerapi/packages.html	[5]
[35]	JavaRMI	1.41, Mar 24, 1997	Sun Microsystems, Java™ Remote Method Invocation Specification. http://java.sun.com/products/DJ/1.1/docs/guide/rmi/spec/rmiTOC.doc.html	[5]
[36]	Java VM	ISBN 0-201432943	The Java Virtual Machine Specification (2nd edition), T. Lindholm and F. Yellin, Addison-Wesley.	[2]
[37]	JFIF		JPEG File Interchange Format, Eric Hamilton, C-Cube Microsystems. http://www.w3.org/Graphics/JPEG/jfif3.pdf	[5]
[38]	PersonalJAE		Sun Microsystems, PersonalJava Application Environment Specification Version 1.2: http://java.sun.com/products/personaljava/	[5]
[39]	PNG	V1 01-Oct-96	Portable Network Graphics. Available at http://www.w3.org/TR/REC-png.html	[5]
[40]	RFC 1321	April 1992	The MD5 Message-Digest Algorithm	[3]
[41]	RFC 1990	August 1996	(MP) "The PPP Multilink Protocol", K. Sklower, B. Lloyd, G. McGregor, D. Carr, T. Coradetti.	[3]
[42]	RFC 2616	June 1999	IETF Hypertext Transfer Protocol -- HTTP/1.1	[3]

	Reference	Edition	Description	Note
[43]	RFC 2396	August 1998	IETF Uniform Resource Identifiers (URI): Generic Syntax	[3]
[44]	RFC 768	28.08.1980	(UDP) "User Datagram Protocol", J. Postel.	[3]
[45]	RFC 791	01.09.1981	(IP) "Internet Protocol", J. Postel.	[3]
[46]	RFC 793	01.09.1981	(TCP) "Transmission Control Protocol", J. Postel.	[3]
[47]	RFC 1112	August 1989	IETF Host extensions for IP multicasting.	[3]
[48]	TR 101 194	1.1.1	Guidelines for the use of ETS 300 802	
[49]	TR 101 200	1.1.1	"Digital Video Broadcasting (DVB); Guideline for the use of DVB specifications and standards".	
[50]	TR 101 201	1.1.1	DVB Interaction channel for SMATV systems	
[51]	TR 101 202	1.1.1	Guidelines for the use of EN 301 192	
[52]	HAVi		See annex V, "(normative): HAVi Level 2 User Interface" on page 525.	
[53]	Java TV	04.04.2000	Java TV API 1.0 Release Candidate D	
[54]	Hunt, R.W.G.	1987	Measuring Colour, Ellis Horwood Series in Applied Science and Industrial Technology. Ellis Harwood Limited, Chichester, England.	
[55]	ITU-T X.501	1993	ITU-T Recommendation X.501: Information Technology - Open Systems Interconnection - The Directory: Models, 1993.	
[56]	ITU-T X.509	08/97	Information technology - Open Systems Interconnection - The Directory: Authentication framework	
[57]	ITU-T X.520	11/93	Information Technology Open Systems Interconnection The Directory: Selected Attribute Types	
[58]	RFC 2313	March 98	PKCS #1: RSA Encryption Version 1.5	[3]
[59]	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)"	
[60]	RFC 2459	January 1999	Internet X.509 Public Key Infrastructure. Certificate and CRL Profile.	[3]
[61]	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.	
[62]	JSSE	1.0	Java Secure Sockets Extension API Specification http://java.sun.com/products/jsse/doc/apidoc/index.html	
[63]	ETS 300 706	Edition 1 - 1997-05	Enhanced Teletext Specification	[6]
[64]	FIPS-180-1	April 1995	NIST, FIPS PUB 180-1: Secure Hash Standard http://csrc.nist.gov/fips/fip180-1.txt	[4]
[65]	RFC 2246	January 1999	The TLS Protocol, Version 1.0	[3]
[66]	RFC 2045	November 1996	Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies	[3]

3 Definitions and abbreviations

3.1 Definitions

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

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

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

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

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

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

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

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

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

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

character set: See character encoding

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

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 support application (native, plug-in or downloaded). The nature of the process is not defined explicitly as it depends on the nature of the support application itself. More than one such locus of activity may be present in any given support application

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 supporting application 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 of a combination of the two

hHardware 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 if 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 [43], 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 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.

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

viewer: End-user of the MHP terminal

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

3.2 Acronyms and Abbreviations

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

API	Application Programming Interface
AV	Audio Video
AWT	Abstract Windowing Toolkit
CA	Conditional Access
CI	Common Interface
CLUT	Colour Lookup Table

CSS	Cascading Style Sheets
DAVIC	Digital Audio Visual Council
DCT	Discrete Cosine Transformation
DECT	Digital Enhanced Cordless Telecommunications
DOM	Document Object Model
DSM-CC	Digital Storage Media - Command and Control
DSM-CC-OC	Digital Storage Media - Command and Control Object Carousel
DSM-CC-UU	Digital Storage Media - Command and Control User to User
DVB	Digital Video Broadcasting
ECMA	European Computer Manufacturers Association
EPG	Electronic Program Guide
ETSI	European Telecommunications Standards Institute
GIF	Graphics Interchange Format
GSM	Global System for Mobile communications
GUI	Graphical User Interface
HTML	Hyper Text Mark-up Language
HTTP	Hyper Text Transport Protocol
I/O	Input / Output
IHDN	In Home Digital Network
IP	Internet Protocol
IPR	Intellectual Property Rights
IRD	Integrated Receiver Decoder
ISDN	Integrated Services Digital Network
ISO	International Standardisation Organisation
ITU	International Telecommunication Union
JDK	Java Development Kit
JFIF	JPEG File Interchange Format
JMF	Java Media Framework
JPEG	Joint Picture Expert Group
LMDS	Local Multipoint Distribution System
MHEG	Multimedia Hypermedia Expert Group
MHP	Multimedia Home Platform
MMDS	Multipoint Microwave Distribution System
MPEG	Moving Picture Expert Group
OC	Object Carousel
OS	Operating System
OSD	On Screen Display
PFR	Portable Font Resource
PMT	Program Map Table
PNG	Portable Network Graphics
PSI	Program Specific Information
PSTN	Public Switched Telephone Network
RAM	Random Access Memory
ROM	Read Only Memory

SI	Service Information
SMATV	Satellite Master-Antenna Television
TCP	Transmission Control Protocol
TS	Transport Stream
UCS	Universal Multiple-Octet Coded Character Set
UDP	User Datagram Protocol
UI	User Interface
URL	Uniform Resource Locator
UTF	UCS Transformation Coding
UU	User to User
VM	Virtual Machine
WAN	Wide Area Network

4 Conventions

[Editorial remark: if any]

5 Basic Architecture

5.1 Context

At its simplest level, the MHP is set in the following context (see figure 1). 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.

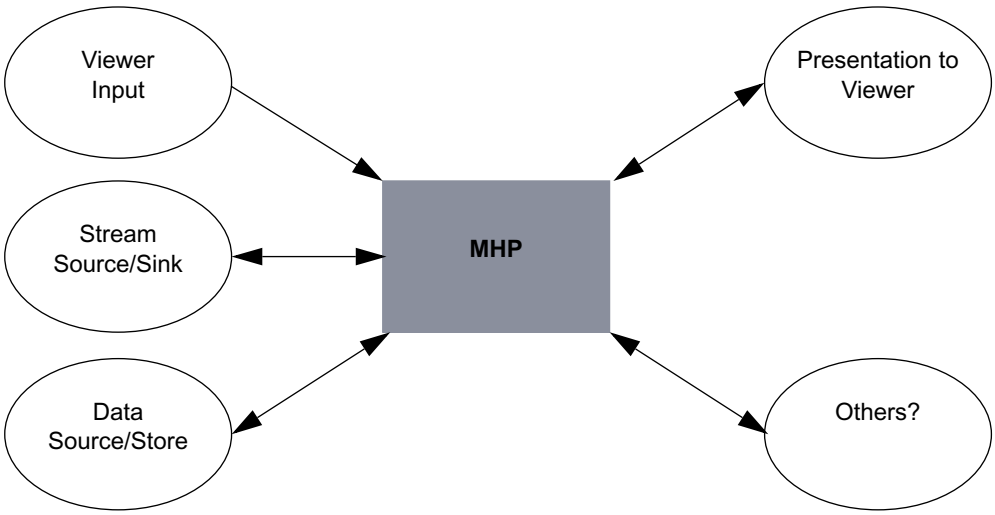


Figure 1 : MHP context

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 2 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.

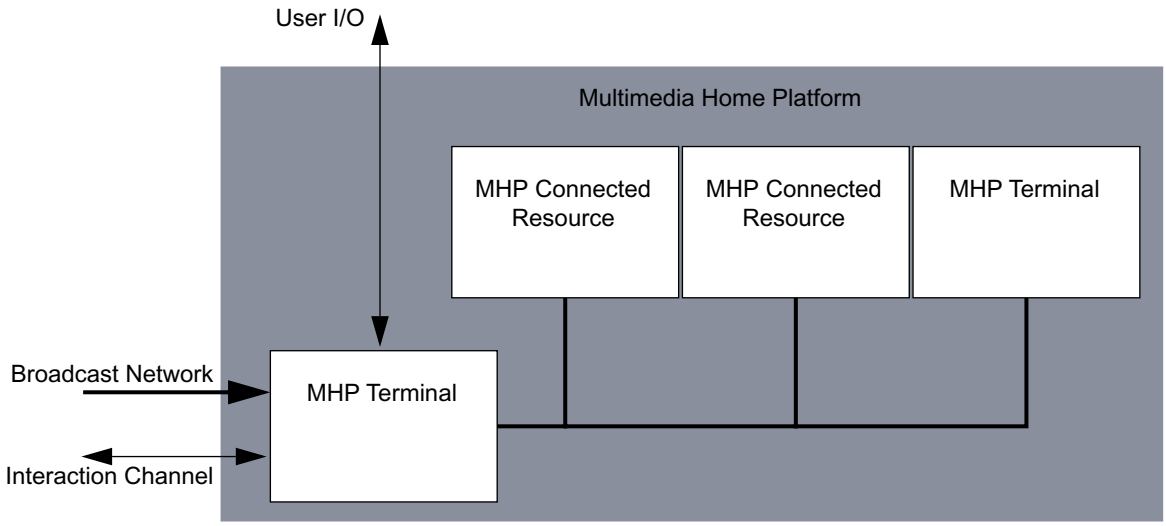


Figure 2 : External interfaces between an MHP and the outside world

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 3):

- Resources
- System software
- Applications

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

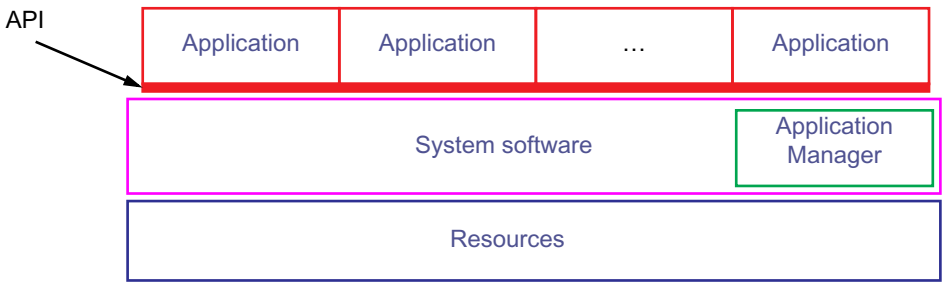


Figure 3 : Basic architecture

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 3 on page 32 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.

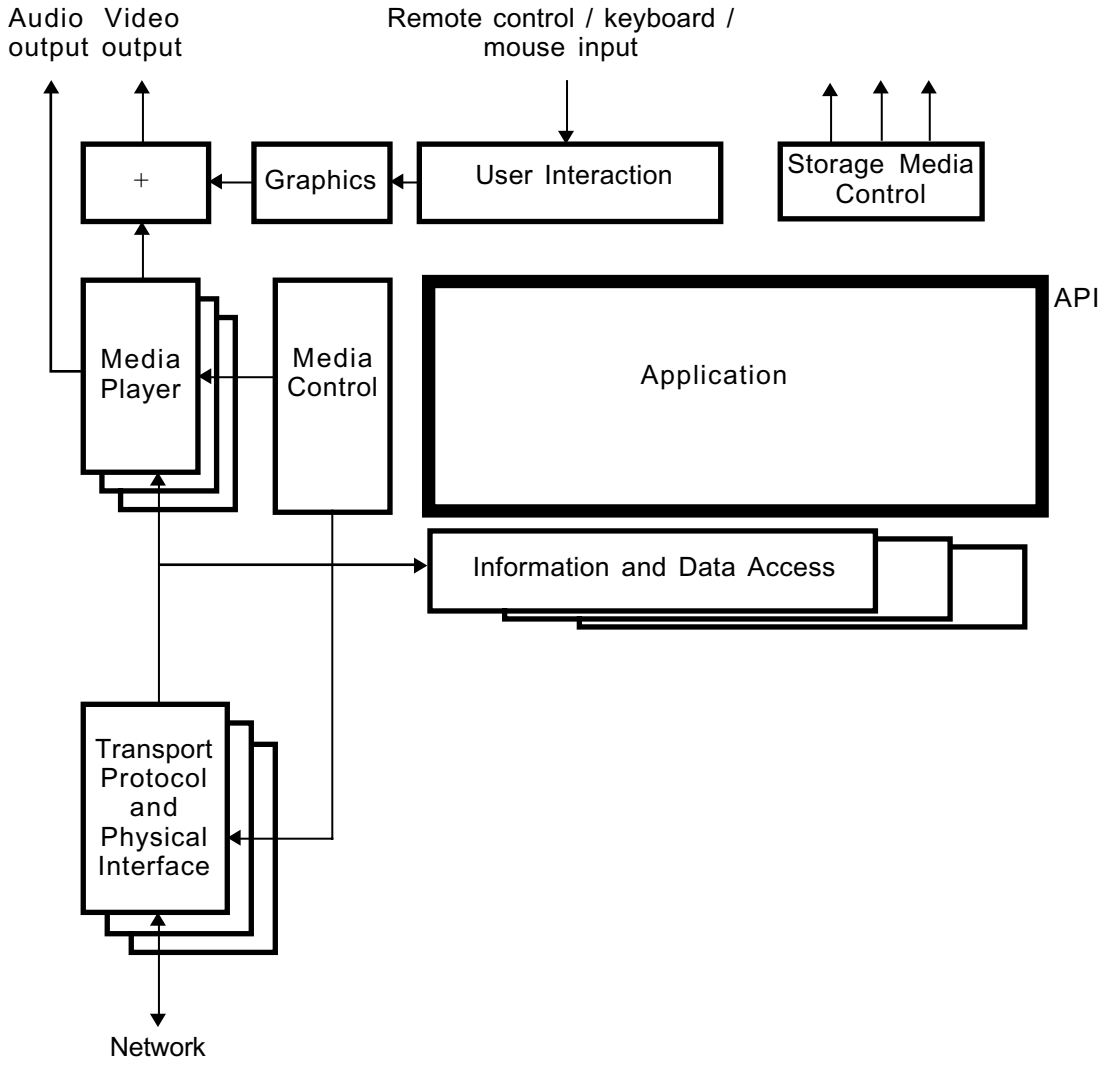


Figure 4 : Interfaces between an MHP application and the MHP system

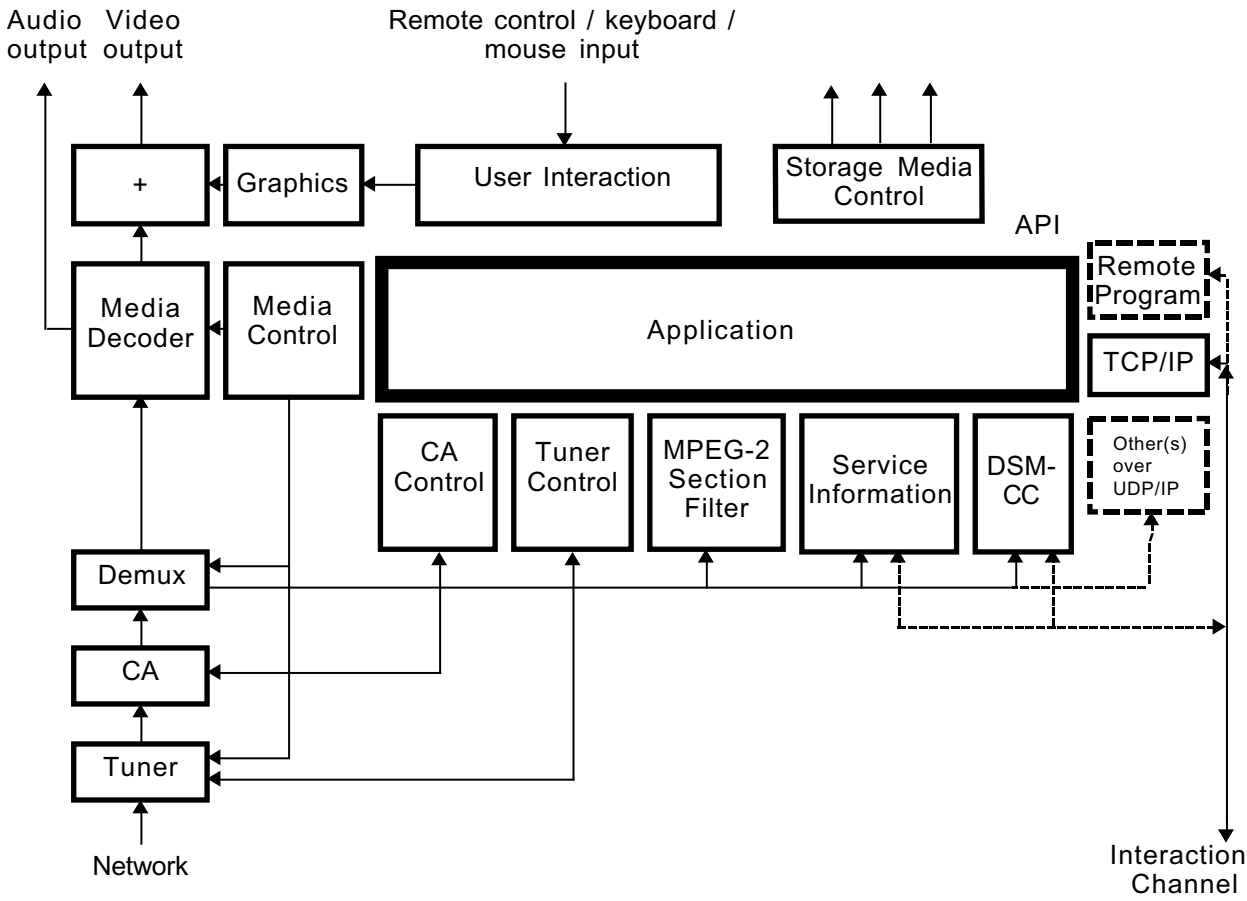


Figure 5 : Interfaces between an MHP application and the MHP system with more details

The diagrams in figure 4 and figure 5 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 4 and figure 5, 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. The motivation of this technology is to enable support of 'legacy' interactive systems.

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 legacy system alone, and be conformable in the legacy system.

Figure 6 illustrates the position of two types of Plug-in in the MHP.

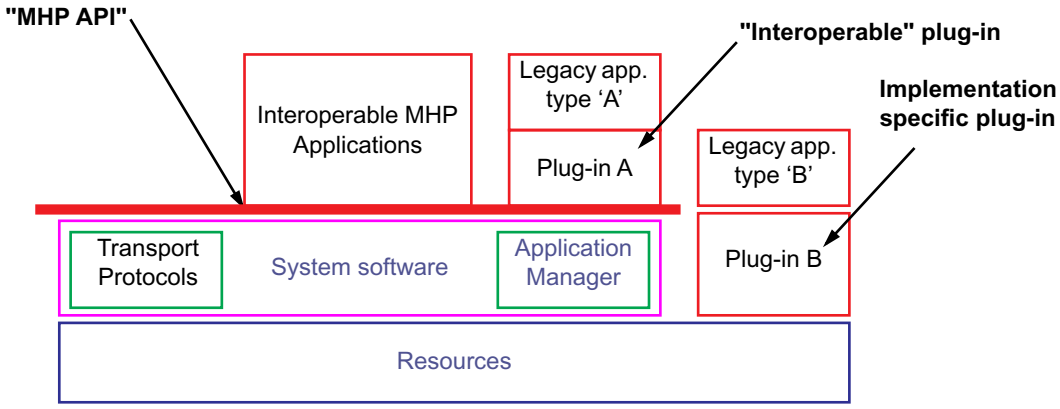


Figure 6 : Illustrative plug-in implementation options

There are two possible types of Plug-in implementation:

- in native code provided or ported for the implementation concerned
- as an MHP application which then may become resident or both

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

5.4.1 Security Model

Plug-ins must have sufficient access to the resources in the platform to be interoperable with the equivalent legacy platform. A native plug-in can have access to many of the resources of the platform irrespective of the MHP security model. The native plug-in itself will then manage the security of the system with respect to the legacy applications that it interprets.

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 [50] respectively for CATV, PSTN/ISDN, DECT, GSM, LMDS and SMATV networks. The network dependent protocols work together with the Network Independent Protocols.

6.2 Broadcast Channel Protocols

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

Other protocols and their APIs are considered as extensions to the DVB MHP platform, see H, "(informative): Extensions" on page 222.

Figure 7 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 163). The full details of the APIs that provide access to these broadcast protocols are in chapter 11, "DVB-J Platform" on page 87.

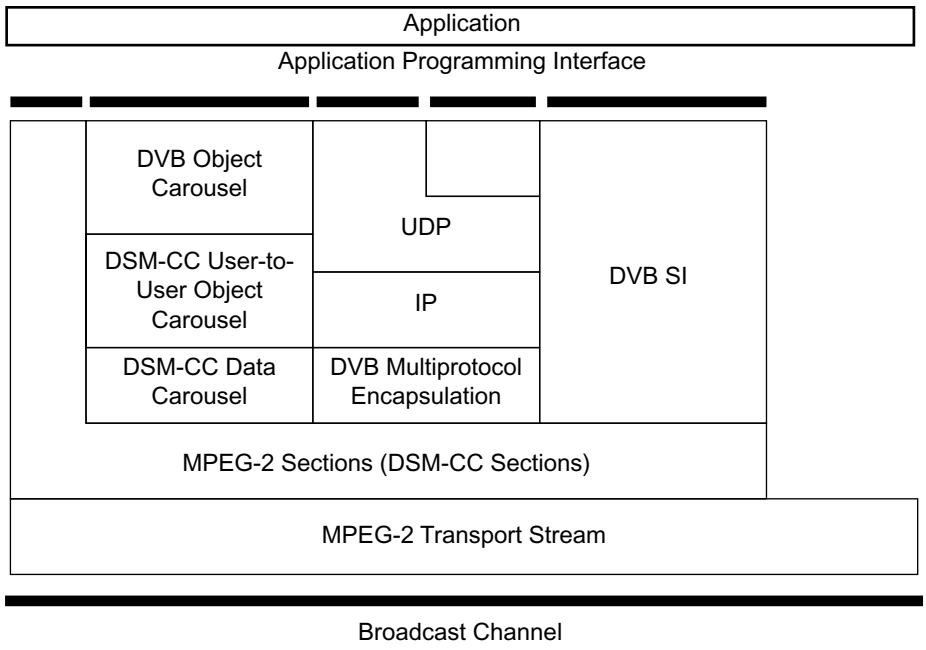


Figure 7 : Broadcast Channel Protocol Stack

6.2.1 MPEG-2 Transport Stream

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

6.2.2 MPEG-2 Sections

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

6.2.3 DSM-CC Private Data

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

6.2.4 DSM-CC Data Carousel

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

6.2.5 DSM-CC User-to-User Object Carousel

DSM-CC User-to-User Object Carousel protocols as defined in ISO/IEC 13818-6 [26] with the restrictions and extensions as defined in EN 301 192 [5], TR 101 202 [51] and annex B, "(normative): Object carousel" on page 168.

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 [36].

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.).

Content formats as specified in 7, "Content formats" on page 41.

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 [45].

6.2.8 User Datagram Protocol (UDP)

User Datagram Protocol as defined in RFC 768 [44].

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 8 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 163). The full details of the APIs that provide access to these broadcast protocols are in chapter 11, "DVB-J Platform" on page 87.

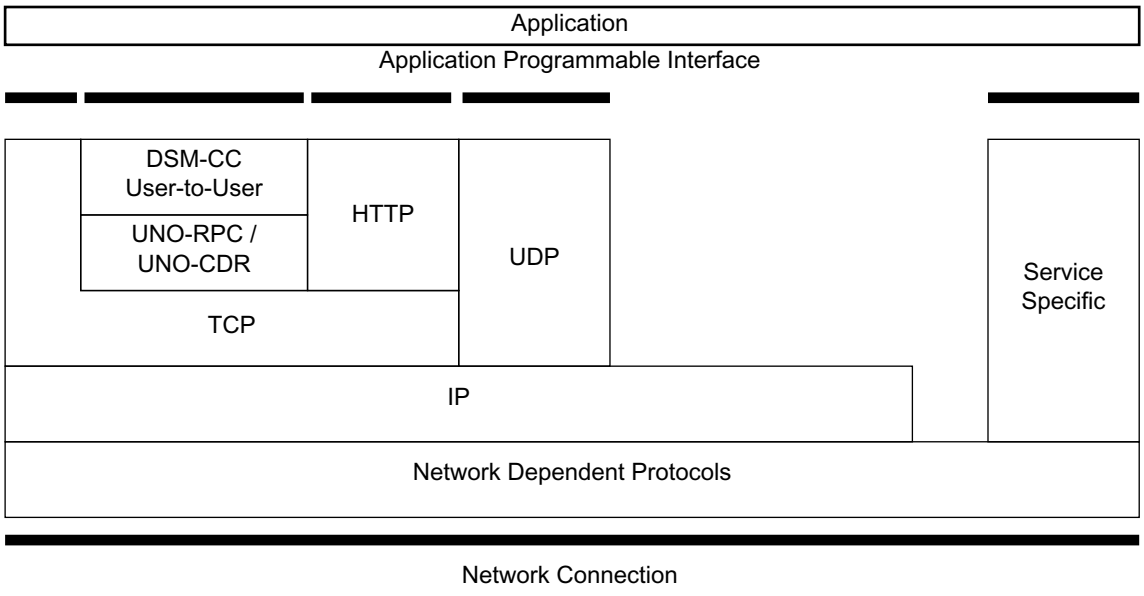


Figure 8 : Interaction Channel Protocol Stack

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 [50] respectively for CATV, PSTN/ISDN, DECT, GSM, LMDS and SMATV networks.

6.3.2 Internet Protocol (IP)

Internet Protocol as defined in RFC 791 [45].

6.3.3 Transmission Control Protocol (TCP)

Transmission Control Protocol as defined in RFC 793 [46].

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 [51].

6.3.7 Hypertext Transfer Protocol (HTTP)

Hypertext Transfer Protocol as defined in RFC 2616 [42].

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 [51]. 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 [44].

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 44.

7.1.1.2 JPEG

JPEG as defined in ISO/IEC 10918-1 [21] using the JFIF [37] 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 [39].

See also 15.1, "PNG - restrictions" on page 165.

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()`.

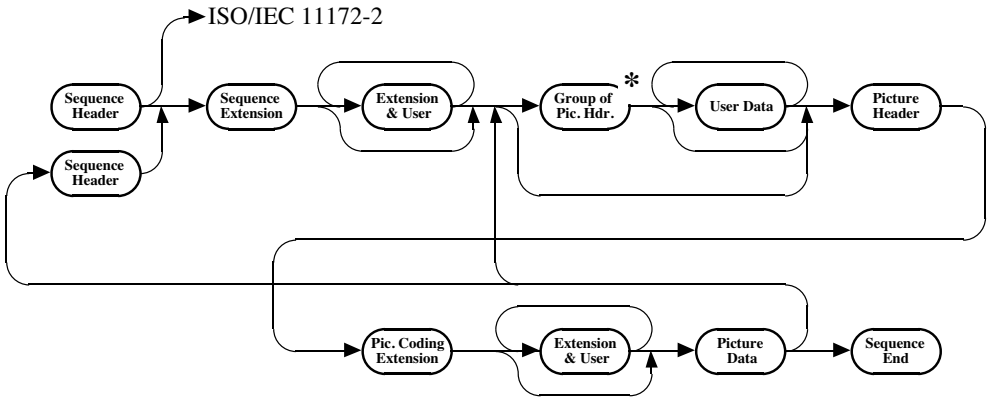
Firstly, the content of each of the chunks of bytes fed to the decoder shall comply with the following syntax:

```
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 9 :

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 340.

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 ETR 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 ETR 154 [9].

7.1.5 Monomedia format for text

Java modified UTF-8 as defined in Java Language Spec. [33] 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 G.4, "Resident fonts" on page 220.

7.2 Broadcast streaming formats

7.2.1 Audio

MPEG Audio with the restrictions and enhancements defined in ETR 154 [9]

7.2.2 Video

Standard Definition 25 Hz MPEG Video with the restrictions and enhancements defined in ETR 154 [9].

7.2.3 Subtitles

The content formats supported for subtitles are:

- DVB Subtitles
- Teletext

See 13.5, "Subtitles" on page 157.

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 [63] 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" on page 220.

See also annex D, "(normative): Text presentation" on page 199.

7.4 Downloadable Fonts

PFR (Portable Font Resource) 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.

See also D.2.2, "Downloaded fonts" on page 199.

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 YC_rC_b colourspace that overlaps with the sRGB gamut).

NOTE: that the presentation of images using colors 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. [54])
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 YC_bC_r transmitted values are computed by similar relationships.

Therefore ITU-R BT.709 YC_bC_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

$R'_{sRGB} = 1.055 * R_{sRGB} \text{ ^}(1.0/2.4) - 0.055$

$G'_{sRGB} = 1.055 * G_{sRGB} \text{ ^}(1.0/2.4) - 0.055$

$B'_{sRGB} = 1.055 * B_{sRGB} \text{ ^}(1.0/2.4) - 0.055$

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 41.
"image/png"	".png"	As defined in 7.1.1.3, "PNG" on page 41 possibly with a constrained profile as defined in 15.1, "PNG - restrictions" on page 165.
"image/gif"	".gif"	As defined in 7.1.1.4, "GIF" on page 41.
"image/mpeg"	".mpg"	As defined in 7.1.2, "MPEG-2 I-Frames" on page 41.
"video/mpeg"	".mpg"	As defined in 7.2.2, "Video" on page 43.
"audio/mpeg"	".mp2"	As defined in 7.1.4, "Monomedia format for audio clips" on page 43 or as defined in 7.2.1, "Audio" on page 43.
"text/dvb.utf8"	".txt"	As defined in 7.1.5, "Monomedia format for text" on page 43.
"image/dvb.subtitle"	".sub"	As defined in 7.2.3, "Subtitles" on page 43.
"text/dvb.subtitle"		
"text/dvb.teletext"	".tlx"	As defined in 7.2.3.2, "Teletext" on page 43.
"application/dvb.pfr"	".pfr"	As defined in 7.4, "Downloadable Fonts" on page 44.
"application/dvbj"	".class"	A DVB-J class file. See 6.2.5.1, "DVB-J class files" on page 38.
"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.

8 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 55.
- How to signal a DVB HTML application in 10, "Application Signalling" on page 63.
- Extensions on how to transport a DVB HTML application in 6.2.5.2, "DVB-HTML document files" on page 38.

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.

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.

MHP terminals may limit on the number of broadcast services which can be presented simultaneously. In this situation, selecting one service may result in stopping the presentation of another service. In the case of MHP terminals which only support the presentation of one service at one time, selecting a new service to be presented will always result in stopping the presentation of the previously presented service.

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 68 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.

9.1.3 Support For Execution of Multiple Simultaneous Applications

The set of applications that are signalled within a service can be presented and executed concurrently.

MHP terminals shall be able to support applications from that set (using the same screen) at least as defined in the minimum platform capabilities section of this specification. MHP terminals are required to support execution of the set of such applications for each broadcast service which they permit to be presented simultaneously.

Broadcasters should ensure that simultaneous running of the set of applications for a service is comprehensible to the user and does not yield perceptible interference problems.

9.1.4 Stopping Applications

MHP applications may stop themselves voluntarily using the MHP APIs or may be stopped by the MHP terminal in a number of situations. Examples of situations where this shall be allowed include:

9.1.4.1 A new service being selected replacing a previously selected one

When a new service is selected and replaces a previously selected one, applications from the former service shall only continue to execute where they are signalled in the new service. If an application is not signalled in that signalling then it will be stopped by the MHP terminal. Where an application is known to be bound to a single service, the broadcaster can identify that application as service bound using the service_bound_flag in the application descriptor. Such applications shall be stopped as soon as possible by the MHP terminal and without needing the AIT for the new service to be available. This allows the autostart application(s) of the new service to be started earlier than would otherwise be the case.

9.1.4.2 The stopping of an application by another application

Subject to the security policy of an MHP terminal, one application may request to stop another application. In such a case, the resident Application Manager, after a successful security check, kills the application otherwise that application shall continue running, without interruption..

9.1.4.3 Changes in the application signalling to request a particular application be stopped

The broadcaster may request an MHP terminal to stop an application using the control codes in the AIT. The precise semantics of these are dependent on the application format.

9.1.4.4 Stopping by the MHP terminal due to a shortage of resources

Where an MHP terminal has insufficient resources to continue the execution of one of the running applications, the MHP terminal is allowed to decide to stop an application without user intervention. When this happens the MHP shall first kill the applications by increasing order of the priority fields signalled in the application_priority field in the application descriptor for each application.

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.

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,
- only a single application with a given Application identification is allowed to run at any time,
- the application is a newly introduced autostart application or has newly been given autostart status.

So:

- when a service is selected the receiver shall launch at most one instance of each autostart application that it can support,
- if after service selection an autostart application that the receiver can support is introduced or a previously listed supportable application gains autostart status then the application shall be launched subject to normal resource limitations, etc.

However, if an autostart application terminates it is not restarted unless it again becomes a new autostart application. An application becomes a new autostart application in the following cases:

- The receiver navigates away from the service and then selects a service where the application is autostart.
- The application is removed from the AIT and then is re-introduced.
- The autostart status of the application is reset then set again.

NOTE 1: In summary the autostart status of an application is in effect an edge trigger rather than level trigger signal.

NOTE 2: These semantics for the autostart behaviour address "de-bouncing" the case where an autostart application terminates voluntarily. They do not address the case where the receiver terminates the application.

NOTE 3: This specification does not describe in detail the timing required for the broadcast signalling to renew the autostart status of an application.

9.2 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 456 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 [53]. 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 `notifyDestroyd()` method on the `javax.tv.xlet.XletContext` interface. This interface also includes other methods to allow a DVB-J application to request or notify changes in its state.

The application listing and launching API allows an application to indirectly control the lifecycle of another application subject to security policy. This control is indirect because an application cannot invoke an Xlet state method directly, but goes through this API. This ensures that the resident Application Manager can always keep track of all the application that are running.

When a DVB-J application is stopped by an MHP terminal, the `destroyXlet` method of the signalled Java class implementing the Xlet interface, i.e. the start of the application shall be called by the application manager. In the case of the application being stopped due to a service selection operation, the stopping of the application shall be unconditional. This method call gives applications their last opportunity to save state before their execution stops. Applications which wish to survive the user of the MHP terminal zapping away from their service (e.g. during an advertising break) must save their state and reload that state when they are re-started if the user returns to that service later.

9.2.3 DVB-J 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.

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 or the streamed media 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 10.4.3, "Visibility of AIT" on page 65. The service being presented in the service context shall not be changed by usage of these APIs.

9.2.4 DVB-J Application Lifecycle

9.2.4.1 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.

9.2.4.2 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 10 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 in the AIT (see 10.4, "Application Information Table" on page 65)).

  • 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 98).

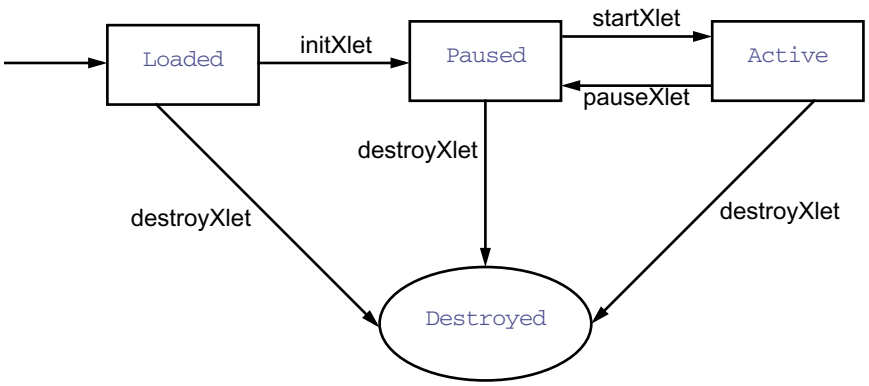


Figure 10 : Xlet lifecycle state machine diagram

Table 5 : Valid DVB-J application lifecycle states

State Name	Description
`Loaded`	The DVB-J application has been loaded and has not been initialized. The signalled Java class used to initiate a DVB-J application must implement the `javax.tv.xlet.Xlet` interface. Otherwise, the class (and hence the application) may be ignored. An instance of the signalled Java class is created by the application manager using the `new` operator. Therefore a DVB-J application must have a public "default constructor". Otherwise, the class (and hence the application) will be ignored. If the default constructor returns without throwing an uncaught exception, then the application is considered to be "Loaded", otherwise the application immediately enters the `Destroyed` state and is discarded. Once the application has been successfully loaded and instantiated, the application manager can transition the application to the `Paused` state by invoking the `initXlet` method on the signalled class (implementing the Xlet interface). **Notes:** • Application initialisation is intended to occur in the `initXlet` method, rather than in the default constructor. • This state is entered only once per instance of an DVB-J application.
`Paused`	A `Paused` 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 `Active` state. This state is entered: • From the `Loaded` state after the `Xlet.initXlet()` method returns successfully when invoked by the application manager (the first time). Other invocations of this method do not cause the change of state. Note that the application manager should only call `initXlet()` once per instance of a DVB-J application. • From the `Active` state after the `Xlet.pauseXlet()` method returns successfully when invoked by the application manager. Other invocations of this method do not cause the change of state. • From the `Active` state upon entering the `XletContext.notifyPaused()` method.
`Active`	The DVB-J application is functioning normally and providing service. This state is entered: • From the `Paused` after the `Xlet.startXlet()` method returns successfully.
`Destroyed`	The DVB-J application has released all of its resources and terminated. This state is entered: • When the Xlet's `destroyXlet()` method returns successfully. The `destroyXlet()` method shall release all resources held and perform any necessary clean up so it may be garbage collected. • Upon entering the `XletContext.notifyDestroyed()` method. The Xlet performs its clean up actions before calling the `notifyDestroyed()` method. **Note:** This state is only entered once.

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 `Loaded` 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 `Paused` 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 `Active` 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.

9.2.5 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 97.

9.2.5.1 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.

9.2.6 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.

9.2.6.1 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 98.

9.2.6.2 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 89.

9.2.6.3 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 99 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.

9.2.6.4 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 87.

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 Support application

A support application 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 support application (native, plug-in or downloaded). The nature of the process is not defined explicitly as it depends on the nature of the support application itself. More than one such locus of activity may be present in any given support application.

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.

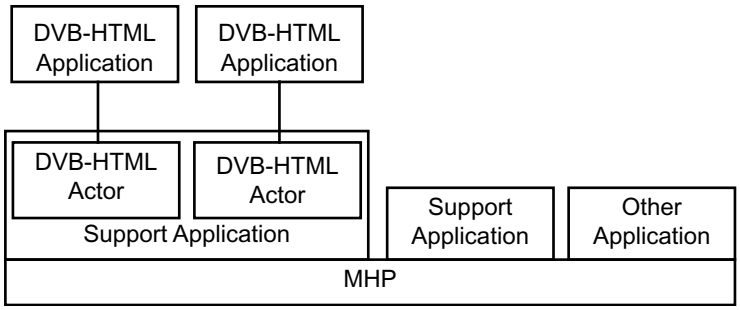


Figure 11 : Relationships between actors and applications

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 DVB-HTML documents making up a DVB-HTML application is defined by a regular expression over the locator language, broadly a locator consists of a text string in the following form RFC 2396 [43] 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 [61] section 2.8.4.:

Relative locators in the DVB-HTML application are expanded to a full URI as defined in RFC 2396 [43], (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 84). 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 supporting application 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 63.

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 support application. It can also at this point begin pre-fetching material.

If the application manager is unable to instantiate a support application either through lack of resources, or no suitable support application 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 support application 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.

9.3.2.3 Application Launching

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.

When a DVB-HTML application transitions to a document that is not signalled as within the current DVB-HTML application boundary, then a new DVB-HTML actor for the destination document may be created if the link is to another DVB-HTML application that is being signalled as OK to run.

The application manager handles the new document to be displayed in the same way as starting a freshly signalled DVB-HTML application.

Depending on the attributes of the link and the capabilities of the support applications, the new actor may run concurrently with the launching DVB-HTML application (e.g. in a new window), or it may temporarily replace the launching DVB-HTML application.

Creation of the new actor may require that the calling DVB-HTML application be suspended if it will have no visible output on the screen after the transition, otherwise the calling DVB-HTML application will simply lose the focus.

9.3.2.4 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 68) and is an abstract view of how a DVB-HTML application operates, and considers the kinds of resources that a support application 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 support application;

9.3.2.4.1 State diagram

The following transition diagram summarizes the states and the transitions between them.

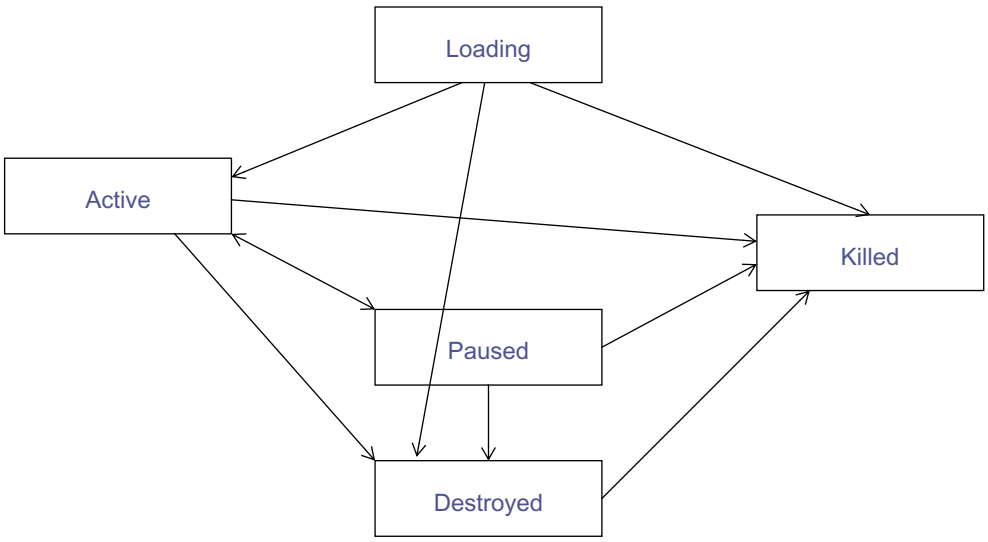


Figure 12 : DVB-HTML application life cycle state diagram

9.4 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 support application 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.

9.4.1 Loading

9.4.1.1 Name:

Loading

9.4.1.2 Entry actions:

Instantiation of an actor.

9.4.1.3 Activities:

Waiting for documents to be available and loading documents without rendering them.

9.4.1.4 Resources:

Content, signalling, Output

9.4.1.5 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,

9.4.1.6 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 support application 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.

9.4.2 Active

9.4.2.1 Name:

Active

9.4.2.2 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.

9.4.2.3 Entry actions

If application is signalled as pre-fetch wait for trigger before displaying anything.

9.4.2.4 Resources:

Content, signalling, output, input.

9.4.2.5 Transitions:

Pause.

- If the support application 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

9.4.2.6 Comment:

This state is the steady state,

In this state it support application 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.

9.4.3 Paused

9.4.3.1 Name:

Paused

9.4.3.2 Activities:

DVB-HTML actor should minimise its use of resources.

9.4.3.3 Resources:

Application specific.

9.4.3.4 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

9.4.3.5 Comment:

The semantics of this state are both support application 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.

9.4.4 Destroyed

9.4.4.1 Name:

Destroyed

9.4.4.2 Activities:

Loading documents. Rendering documents. Consuming events. Interact with the user.

9.4.4.3 Resources:

input and output.

9.4.4.4 Transitions:

Killed.

- {If the DVB-HTML application is signalled as KILL} OR {local event forces the actor to terminate, possibly through application manager}

9.4.4.5 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 support application 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.

9.4.5 Killed

9.4.5.1 Name:

Killed

9.4.5.2 Entry actions:

Release of resources.

9.4.5.3 Activities:

Termination of the DVB-HTML application.

9.4.5.4 Resources:

none.

9.4.5.5 Transitions:

none

9.4.5.6 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.

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 25

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 26.

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 24, "Semantic of selector bytes" on page 76
- 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 81
- Define the application type specific life cycle control codes in 10.6, "Control of application life cycle" on page 68.

Where constant values are registered by this specification extend the table 37, "Registry of constant values" on page 85.

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 13818-1private 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 65).

Optionally the PMT may include Data broadcast id descriptors.

NOTE: Inclusion of Data broadcast id descriptors enables receivers to start mounting the file system that delivers applications concurrently with acquiring the AIT that identifies which applications are of interest. Enabling this concurrent operation may allow receivers to accelerate their activation of an interactive application. See B.2.8, "Mounting an Object Carousel" on page 191.

The Data broadcast id descriptor identifies the "principal" component of the data broadcast. The detailed semantics of this optional signalling reflects the transport protocol. For example, in the case of a DVB Object Carousel it identifies the component carrying the DSI.

There may also be certain protocol specific descriptors in the PMT. For example, the Object Carousel requires the inclusion of the carousel_id_descriptor (see B.2.8, "Mounting an Object Carousel" on page 191).

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 71.

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.

The minimum repetition rate for each AIT subtable is 10 seconds.

10.4.1 Default AIT monitoring

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.

10.4.2 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 70.

10.4.3 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.

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.

10.4.4 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.

10.4.5 Syntax of the AIT

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 7 : Application Information Section syntax

	No.of Bits	Identifier
application_information_section() {		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
application_type	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
reserved_future_use	4	bslbf
common_descriptors_length	12	uimsbf
for(i=0;i<N;i++){		
descriptor()		
}		
reserved_future_use	4	bslbf
application_loop_length	12	uimsbf
for(i=0;i<N;i++){		
application_identifier()		
application_control_code	8	uimsbf
reserved_future_use	4	bslbf
application_descriptors_loop_length	12	uimsbf
for(j=0;j<N;j++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

table_id: This 8 bit integer with value 0x74 identifies this table.

section_syntax_indicator: The section_syntax_indicator is a 1-bit field which shall be set to "1".

section_length: This is a 12-bit field, the first two bits of which shall be '00'. The remaining 10 bits specify the number of bytes of the section starting immediately following the section_length field, and including the CRC_32. The value in this field shall not exceed 1021 (0x3FD).

application_type: This is a 16-bit field which identifies the type of the applications described in this AIT sub_table. See table 8.

Table 8 : Application types

application_type	description
0x0000	reserved
0x0001	DVB-J application
0x0002	DVB-HTML application
0x0003…0xFFFF	subject to registration with DVB

version_number: This 5-bit field is the version number of the sub_table. The version_number shall be incremented by 1 when a change in the information carried within the sub_table occurs. When it reaches value "31", it wraps around to "0". When the current_next_indicator is set to "1", then the version_number shall be that of the currently applicable sub_table. When the current_next_indicator is set to "0", then the version_number shall be that of the next applicable sub_table.

current_next_indicator: This 1-bit indicator, when set to "1" indicates that the sub_table is the currently applicable sub_table. When the bit is set to "0", it indicates that the sub_table sent is not yet applicable and shall be the next sub_table to be valid.

section_number: This 8-bit field gives the number of the section. The section_number of the first section in the sub_table shall be "0x00". The section_number shall be incremented by 1 with each additional section with the same table_id, and application_type.

last_section_number: This 8-bit field specifies the number of the last section (that is, the section with the highest section_number) of the sub_table of which this section is part.

common_descriptors_length: This 12-bit field gives the total length in bytes of the following descriptors. The descriptors in this descriptor loop apply for all of the applications contained in this AIT sub_table.

application_control_code: This 8-bit field controls the state of the application. The semantics of this field is application type dependant. See 10.6, "Control of application life cycle" on page 68.

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 68.

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.

10.4.6 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.

10.5 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 9 : Application identifier syntax

	No.of Bits	Identifier	Value
application_identifier {			
organisation_id	32	bslbf	
application_id	16	bslbf	
}			

organisation_id: This 32 bit field is a globally unique value identifying the organisation that is responsible for the application. These values are registered in ETR 162 [10].

This field is reproduced in the organisationName field of the subject name in the "leaf" certificate of an authenticated application (see 12.5.6, "subject" on page 113).

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.

When the value of the application id field is 0xFFFF this application identifier matches all applications with the same organisation_id. This value shall not be used to identify an application but, for example, this use is allowed in the External application authorisation descriptor see 10.7.5 on page 75.

10.5.2 Effects on life cycle

The main concepts here are:

- On service change, currently running, previously broadcast, applications are allowed (subject to resource restrictions) to 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 are allowed (subject to resource restrictions) to 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.

10.5.3 Authentication of application identification

See 12.5.6, "subject" on page 113.

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 49.

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.

10.6.2.1 DVB-J

The application control codes for DVB-J applications are listed in 10.

Table 10 : DVB-J application control code values

code	identifier	semantics
0x00		reserved
0x01	AUTOSTART	The 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 **`unconditional`** parameter set to **false**) 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 **`unconditional`** parameter set to **true**) by the application manager.
0x05	reserved	
0x06	REMOTE	This identifies a remote application that is only launchable after tuning.
0x07…0xFF		reserved for future use

See 9.2.4, "DVB-J Application Lifecycle" on page 52.

10.6.2.2 DVB-HTML

The application control codes for DVB-HTML applications are listed in 11.

Table 11 : DVB-HTML application control code values (Sheet 1 of 2)

code	identifier	semantics
0x00		reserved
0x01	AUTOSTART	The Application Entry Point of the DVB-HTML application is loaded. This is loaded into the support application, 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.

Table 11 : DVB-HTML application control code values (Sheet 2 of 2)

code	identifier	semantics
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 tuning.
0x07…0xFF		reserved for future use

See 9.3.2, "DVB-HTML Application Lifecycle" on page 57.

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 10.4.2, "Optimised AIT signalling" on page 65.

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 Default AIT monitoring applies, see 10.4.1, "Default AIT monitoring" on page 65.

Table 12 : application signalling descriptor syntax

	No.of Bits	Identifier
`application_signalling_descriptor() {`		
`  descriptor_tag`	8	uimsbf
`  descriptor_length`	8	uimsbf
`  for( i=0; i<N; i++ ){`		
`  application_type`	16	uimsbf
`  reserved_future_use`	3	bslbf
`  AIT_version_number`	5	uimsbf
`  }`		
`}`		

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 13). Where no 'id specific data' is provided the descriptor just identifies the 'principal' component of a data broadcast.

Table 13 : generic data broadcast id descriptor syntax

	No.of Bits	Identifier	Value
data_broadcast_id_descriptor() {			
descriptor_tag	8	uimsbf	
descriptor_length	8	uimsbf	
data_broadcast_id	16	uimsbf	
for (i=0; i<N; i++) {			
id specific data	8	bslbf	
}			
}			

10.7.2.2 MHP data broadcast id descriptor

When the data broadcast id is one of those defined by this specification (see table 37) the syntax of the data broadcast id descriptor is as shown in table 14. This allows the data broadcast id descriptor to provide information about the autostart applications that exist within the data broadcast.

Table 14 : MHP data broadcast id descriptor syntax

	No.of Bits	Identifier	Value
data_broadcast_id_descriptor() {			
descriptor_tag	8	uimsbf	
descriptor_length	8	uimsbf	
data_broadcast_id	16	uimsbf	
for (i=0; i<N; i++) {			
application_type	16	uimsbf	
}			
}			

descriptor_tag: This 8 bit integer with value 0x66 identifies this descriptor.

data_broadcast_id: This 16 bit field indicates the format of the data broadcast transport protocol. These values are registered in ETR 162.

application_type: This 16 bit field indicates the type of the application (i.e. the engine or plug-in on which the application can be executed). See table 8 on page 67.

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 15 : application descriptor syntax

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

descriptor_tag: This 8 bit integer with value 0x00 identifies this descriptor.

application_profiles_length: This 8-bit field indicates the length of the application_profile loop in bytes.

application_profile: This 16 bit field is an integer value which represents the application type specific profile. This indicates that a receiver implementing one of the profiles listed in this loop is capable of executing the application.

version.major: This 8 bit field carries the numeric value of the major sub-field of the profile version number.

version.minor: This 8 bit field carries the numeric value of the minor sub-field of the profile version number.

version.micro: This 8 bit field carries the numeric value of the micro sub-field of the profile version number.

The four above fields indicate the minimum profile on which an application will run. Applications may test for features found in higher (backwards compatible) profiles and exploit them. The MHP terminal may launch applications where the following expression is true for at least one of the signalled profiles:

(application_profile ∈ terminal_profiles_set)
∧ {(application_version.major < terminal_version.major(application_profile))
∨ [(application_version.major = terminal_version.major(application_profile))
∧ ({application_version.minor < terminal_version.minor(application_profile)}
∨ {[application_version.minor = terminal_version.minor(application_profile)]
∧ [application_version.micro ≤ terminal_version.micro(application_profile)]})]}

Where:
∈ represents "belongs to the set of"
∧ represents "logical AND"
∨ represents "logical OR"

See table 6, "Profile encoding" on page 166 for the encoding of these values.

service_bound_flag: If this field is set to '1', the application is only associated with the current service and so the process of killing the application shall start at the beginning of the service change regardless of the contents of the destination AIT.

visibility: This 2 bit field specifies the intended visibility of the application for possible presentation to the user for possible launching. Table 16 lists the allowed values of this field.

Table 16 : Definition of visibility states for applications

visibility	description
00	This application should 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 should not be visible to users but is visible to applications via an application listing API.
10	Reserved
11	This application can be visible to users and can be visible to applications via an application listing API.

application_priority: This field identifies a relative priority between the applications signalled in this service.

- Where there is more than one application with the same Application identification this priority shall be used to determine which application is started.
- Where there are insufficient resources to continue running a set of applications, this priority shall be used to determine which applications to stop or pause.

transport_protocol_label: This 8-bit field identifies a transport protocol that delivers the application. See transport_protocol_label in Transport protocol descriptor.

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 17 : application name descriptor syntax

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

descriptor_tag: This 8 bit integer with value 0x01 identifies this descriptor.

ISO_639_language_code: This 24-bit field contains the ISO 639.2 [19] three character language code of the language of the following bouquet name. Both ISO 639.2/B and ISO 639.2/T may be used.

Each character is coded into 8 bits according to ISO 8859 [20] and inserted in order into the 24-bit field.

application_name_length: This 8 bit unsigned integer specifies the number of bytes in the application name.

application_name_char: This field caries 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 165.

Table 18 : application icons descriptor syntax

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

descriptor_tag: This 8 bit integer with value 0x0B identifies this descriptor.

icon_locator_length: This 8 bit integer specifies the number of characters in the string that prefixes standard icon file name.

icon_locator_byte: This 8 bit value is one byte of the icon locator string.

The icon locator is the first part of the string that specifies the location of the icon files. This is application type dependant. See table 19.

Table 19 : Icon locator semantics

application_type	description
0x0000	reserved
0x0001	For DVB-J this is a path relative to the base directory of the application as defined in 10.9.2, "DVB-J application location descriptor" on page 82.
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 83.
0x0003…0xFFFF	

icon_flags: This 16 bit field carries a value which is the bitwise OR of the flag bits that identify the icons that are provided for the application. The flag bits are defined in table 20.

Table 20 : Definition of different icon flags

Icon flag bits	Description of icon size and pixel aspect ratio
0000 0000 0000 0001	32 x 32 for square pixel display
0000 0000 0000 0010	32 x 32 for broadcast pixels on 4:3 display (note 1)
0000 0000 0000 0100	24 x 32 for broadcast pixels on 16:9 display
0000 0000 0000 1000	64 x 64 for square pixel display
0000 0000 0001 0000	64 x 64 for broadcast pixels on 4:3 display
0000 0000 0010 0000	48 x 64 for broadcast pixels on 16:9 display
0000 0000 0100 0000	128 x 128 for square pixel display
0000 0000 1000 0000	128 x 128 for broadcast pixels on 4:3 display
0000 0001 0000 0000	96 x 128 for broadcast pixels on 16:9 display
xxxx xxx0 0000 0000	Reserved for future standardisation
NOTE 1: approx. 15/16 pixel aspect ratio on 50 Hz system	

The file names for the icon files are encoded in a standard way:

```
filename = icon_locator "dvb.icon." hex_string
hex_string = 4*4hex
hex                 = digit | "A" | "B" | "C" | "D" | "E" | "F" | "a" |
                          "b" | "c" | "d" | "e" | "f"
digit               = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
                          "8" | "9"
```

Where `hex_string` is the 4 digit hexadecimal representation of the value carried by icon_flags.

10.7.5 External application authorisation descriptor

The "common" (first) descriptor loop of the Application Information Table may contain zero or more external_ application_authorisation_descriptors. Each descriptor contains information about external applications that are allowed to continue to run with the applications listed in this Application Information Table sub-table but cannot be launched from this service.

Table 21 : external application authorisation descriptor syntax

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

descriptor_tag: This 8-bit integer with value 0x05 identifies this descriptor.

application_identifier(): This 48-bit field identifies an application. The structure of this field is defined in 10.5, "Application identification" on page 68.

application_priority: This 8-bit integer specifies the priority that this application assumes in the context of the current service.

See application_priority under 10.7.3, "Application descriptor" on page 72.

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 22 : transport protocol descriptor syntax

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

descriptor_tag: This 8 bit integer with value 0x02 identifies this descriptor.

protocol_id: An identifier of the protocol used for carrying the applications. The values of the protocol_id are be registered here and in ETR162.

Table 23 : Protocol_id

protocol_id	description
0x0000	reserved
0x0001	MHP Object Carousel as defined in annex B, "(normative): Object carousel" on page 168.
0x0002	IP via DVB multiprotocol encapsulation as defined in EN 301 192, TR 101 202
0x0003…0xFFFF	Subject to registration in ETR162

transport_protocol_label: Thus 8 bit field uniquely identifies a transport protocol within this AIT section. The Application descriptor refers to this value to identify a transport connection that carries the application.

selector_byte: Additional protocol specific information.

Table 24 : Semantic of selector bytes

protocol_id	selector byte data
0x0000	reserved
0x0001	See 10.8.1.1 "Transport via OC".
0x0002	See 10.8.1.2 "Transport via IP".
0x0003…0xFFFF	TBD

10.8.1.1 Transport via OC

When the protocol ID is 0x0001 the selector bytes in the Transport protocol descriptor shall be as shown in table 25.

Table 25 : Syntax of selector bytes for OC transport

Syntax	Bits	Mnemonic
remote_connection	1	bslbf
reserved_future_use	7	bslbf
if(remote_connection == '1') {		
original_network_id	16	uimsbf
transport_stream_id	16	uimsbf
service_id	16	uimsbf
}		
component_tag	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 transport stream that is different to the one carrying the AIT. When set the following 3 fields (original_network_id, transport_stream_id and service_id) are included in the selector bytes.

This identifies a remote application that is only launchable after tuning. Such applications shall not be autostarted by receivers but may be visible via an application listing API for possible launching by service selection (but not via an application launching API).

Applications with this flag set shall have their application control code set to REMOTE (see table 10 on page 69 and 11 on page 69).

See 11.7.2, "Application discovery and launching APIs" on page 98.

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 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 26.

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 for the enhanced broadcast and Internet profiles.

Table 26 : Syntax of selector bytes for IP transport

Syntax	Bits	Mnemonic
remote_connection	1	bslbf
reserved_future_use	7	bslbf
if(remote_connection == '1') {		
original_network_id	16	uimsbf
transport_stream_id	16	uimsbf
service_id	16	uimsbf
}		
alignment_indicator	1	bslbf
reserved_future_use	7	bslbf

Table 26 : Syntax of selector bytes for IP transport

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

10.8.1.2.1 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 77.

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 [43].

For URL using the "server" field including the host:port notation as defined in RFC 2396 [43], 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 [47].

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 27 : 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 28 : Syntax of the IPv6 routing descriptor

Syntax	Bits	Mnemonic
routing_descriptor_ip6 () {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for(i=0; i<N; i++) {		
component_tag	8	uimsbf
address	128	uimsbf
port_number	16	uimsbf
address_mask	128	uimsbf
}		
}		

descriptor_tag: This 8-bit field with value 0x07 identifies this descriptor.

descriptor_length: This 8-bit field identifies the number of bytes following the length field.

component_tag: This 8-bit field identifies the service component that carries multicast IP packets for the specified IP address set and port.

An IP address is carried in this service component if the result of AND-ing that address with the address_mask field is equal to the address field.

address: This 128-bit field carries an IPv6 IP address.

port_number: This 16-bit field carries a port number. The value zero is a 'wild card' value and indicates any port number.

address_mask: This 128-bit field carries a mask to be applied to the address field.

10.8.3 Pre-fetch signalling

10.8.3.1 Introduction

This signalling is defined to enable implementations to start fetching files that will be required during the early part of an application's life. Later in an applications' life it can actively request file pre-fetching using API mechanisms. Descriptors in this section do not have a relation to the API-based pre-fetching for this version of this specification.

This signalling is optional to broadcast and optional for implementations to consider.

10.8.3.2 Pre-fetch descriptor

Zero or one pre-fetch descriptors can be included in the "application" (inner) descriptor loop of the AIT. It is defined for use where the protocol_id of the transport is 0x0001 (MHP Object Carousel). Each descriptor is associated with a specific Transport protocol descriptor via the transport_protocol_label.

MHP terminals may use this descriptor to improve application start-up time by pre-fetching modules that have the indicated labels (see "Label descriptor" on page 171).

Table 29 : Syntax of the pre-fetch descriptor

Syntax	Bits	Mnemonic
prefetch_descriptor () {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
transport_protocol_label	8	uimsbf
for(i=0; i<N; i++) {		
label_length		
for(j=0; j<label_length; j++) {		
label_char	8	uimsbf
}		
prefetch_priority	8	uimsbf
}		
}		

descriptor_tag: This 8-bit field with value 0x0C identifies this descriptor.

descriptor_length: This 8-bit field identifies the number of bytes following the descriptor length field.

transport_protocol_label: This 8-bit field identifies the Transport protocol descriptor that specifies the transport connection that delivers the modules to which this prefetch descriptor refers. See "transport_protocol_label" on page 73.

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 171).

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 80), 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 30 : Syntax of the DII location descriptor

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

descriptor_tag: This 8-bit field with value 0x0D identifies this descriptor.

descriptor_length: This 8-bit field identifies the number of bytes following the descriptor length field.

transport_protocol_label: This 8-bit field identifies the Transport protocol descriptor that specifies the transport connection that delivers the modules to which this prefetch descriptor refers. See "transport_protocol_label" on page 73.

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 189.

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 31 : DVB-J application descriptor syntax

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

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 32 : 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.

base_directory_byte: These bytes contain a string specifying a directory name starting from the root of the carousel 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 carousel 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 carousel that is the class implementing the Xlet interface.

This string is a DVB-J class name that is found in the class path (e.g. "com.broadcaster.appA.MainClass").

10.10 DVB-HTML Specific descriptors

10.10.1 DVB-HTML application descriptor

One instance of this descriptor shall be contained in the "application" (inner) descriptor loop of the AIT for each DVB-HTML application. It indicates the value of the application parameters and signals the control applied by the broadcaster on the state of the application.

Table 33 : DVB-HTML application descriptor syntax

	No.of Bits	Identifier	Value
dvb_html_application_descriptor(){			
descriptor_tag	8	uimsbf	
descriptor_length	8	uimsbf	
for(j=0; j<N; j++) {			
parameter_bytes	8	uimsbf	
}			
}			

descriptor_tag: This 8 bit integer with value 0x08 identifies this descriptor.

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 34 : DVB-HTML application location descriptor syntax

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

descriptor_tag: This 8 bit integer with value 0x09 identifies this descriptor.

physical_root_length: This 8 bit integer specifies the length of the physical_root_byte string.

physical_root_bytes: These bytes contain a string specifying the physical root of the application entry point. The semantic of this string is transport protocol specific as shown in table 35 "Transport specific semantic of physical root bytes".

Table 35 : Transport specific semantic of physical root bytes

protocol_id	semantic
0x0000	reserved
0x0001	A directory specification
0x0002	One of the base URLs defined in the Transport protocol descriptor signalled for the application (see 10.8.1.2, "Transport via IP" on page 77).
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 an object carousel 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 36 : DVB-HTML application boundary descriptor syntax

	No.of Bits	Identifier	Value
`dvb_html_application_boundary_descriptor {`			
`descriptor_tag`	8	uimsbf	
`descriptor_length`	8	uimsbf	
`label_length`	8	uimsbf	N1
`for(i=0; i<N1; i++) {`			
`label_bytes`	8	uimsbf	
`}`			

Table 36 : DVB-HTML application boundary descriptor syntax

`for(i=0; i<N; i++) {`			
`  regular_expression_bytes`	8	uimsbf	
`}`			
`}`			

descriptor_tag: This 8 bit integer with value 0x09 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 56.

10.11 Constant values

Table 37 : 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	
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		

Table 37 : Registry of constant values

Where used	Type	Value	Where Defined	Scope
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		0x0E-0x5E		
private data specifier descriptor (note 2)		0x5F		
reserved to MHP for future		0x60-0x7F		
User defined (note 3)		0x80-0xFE		
MHP Object Carousel	data broadcast id	0x00F0	PMT, AIT	SI
reserved for MHP Multi Protocol Encapsulation		0x00F1		
reserved to MHP use		0x00F0 - 0x00FE	PMT, AIT	SI
MHP Application Service	service type	0x10	SDT	SI

NOTE 1: Strictly MessageSubHeader::ObjectInfo in the file message and the bound object info in a file binding of a directory or service gateway message.

NOTE 2: The DVB SI private data specifier descriptor is defined for use in the Application Information Table to introduce private descriptors.

NOTE 3: All user defined descriptors shall be within the scope of a private data specifier descriptor (see 10.4.6, "Use of private descriptors in the AIT" on page 67).

11 DVB-J Platform

11.1 The Virtual Machine

The DVB-J virtual machine is defined in Java VM [36].

11.2 General issues

11.2.1 Basic Considerations

Interoperable application shall not use any classes or methods marked as deprecated in the API specifications referenced or included in this specification.

A DVB-J application is considered to logically run in its own virtual machine instance. For this reason, it cannot rely on finalizers that are defined in application classes being run when the application terminates. When the application manager terminates the entity that represents the virtual machine in which the application is run, a conformant implementation is permitted to not run application finalizers, as spelled out in section 2.17.9 of the Java Language Spec. [33].

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.

11.2.2 Approach to Subsetting

Where a class included in this specification has signature dependencies on classes not in this specification, `java.lang.NoSuchMethodError` and `java.lang.NoClassDefError` will be used if applications attempt to access the methods with the signature dependencies concerned, and this attempt fails. The semantics for these errors defined in Java Class Libraries Vol. 1 [31] will be respected.

Where this specification subsets a package, inclusion of the complete package is allowed but clearly not required. The behaviour of the additional features is not specified for broadcast applications.

11.2.3 Class Loading

The DVB-J application environment shall be written such that each application appears to run within its own classloader or classloader hierarchy for all classes that are not a part of the platform. As a consequence, two applications will never be able to access the same copy of any application-defined static variable.

Signed applications shall only load classes signed by the same leaf certificate.

See 12, "Security" on page 106.

11.2.4 Unloading

Class unloading as defined by section 12.8 of Java Language Spec. [33] and section 2.16.8 of Java VM [36] will be supported.

11.3 Fundamental DVB-J APIs

11.3.1 Java platform APIs

The following packages are defined in Java Class Libraries Vol. 1 [31].

11.3.1.1 java.lang package

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

a) The following methods shall not be used by inter-operable applications:

   Runtime.exec(),
   Runtime.load(),
   Runtime.loadLibrary(),
   Runtime.runFinalizersOnExit(),
   System.exit(),
   System.load(),
   System.loadLibrary(),
   System.runFinalizersOnExit(),
   Thread.stop(),
   Thread.suspend(),
   Thread.resume()

b) The following fields shall not be used by inter-operable applications:

   System.in

c) The following classes shall not be used by inter-operable applications:

   java.lang.Process

d) 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.

e) 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()

f) Only the following properties are required to be supported for `System.getProperty()` and `System.getProperties()`:

   file.separator,
   path.separator,
   line.separator,
   persistent.root (see section 11.5.6, "Persistent Storage API" on page 96)

   All these properties are accessible to unauthorised applications.

g) The `System.setProperties()` and `System.setSecurityManager()` methods will always throw an exception when called by downloaded DVB-J applications.

h) The class `SecurityManager` shall be as specified in PersonalJAE [38].

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.

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.

11.3.1.5 java.io

The java.io package is supported.

11.3.1.6 java.net

From the `java.net` package, the java.net.URL and java.net.InetAddress classes and the exceptions `MalformedURLException` and `UnknownHostException` only are supported. Unless included in the platform as part of a specific profile, All the rest of this package is to be non-supported as defined in section 11.2.2, "Approach to Subsetting" on page 87. The signature dependencies from these to the rest of this package are severed as described in 11.2.2, "Approach to Subsetting" on page 87.

11.3.1.7 java.beans

The java.beans package 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 223.

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 227.

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 Java Class Libraries Vol. 2 [32].

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`
- `Event`
- `EventQueue`
- `FlowLayout`
- `Font`
- `FontMetrics`
- `Graphics`
- `GridBagConstraints`
- `GridBagLayout`
- `GridLayout`
- `IllegalComponentStateException`
- `Image`
- `Insets`
- `ItemSelectable`
- `LayoutManager`
- `LayoutManager2`
- `MediaTracker`
- `MenuContainer`
- `Point`
- `Polygon`
- `Rectangle`
- `Shape`

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 87. The same applies for all references to the java.awt.peer package.

In the Toolkit class, the following methods are required:

- `getDefaultToolkit`,
- `getFontList`,
- `getFontMetrics`,
- `sync`,
- `getColorModel`,
- and all the methods relating to images.

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 [52]. The methods `getSystemEventQueue` and `getSystemEventQueueImpl` are supported however each Xlet will have a unique queue not shared with other Xlets or anything else in the system.

All of the java.awt.image package is required. The encoding of image content types for use by java.awt.image are defined in 7.1.1, "Bitmap image formats" on page 41. 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.

11.4.1.2 TV user interface

The packages `org.havi.ui` and `org.havi.ui.event` defined in HAVi [52] shall be supported.

With the exception of the `HSound.load` method, no methods specified in HAVi [52] 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 493 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. [34]) 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 [53]).
- 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 `HGraphicsDevice.getPunchThroughToBackgroundColor` shall not be used by inter-operable applications.

The class `javax.tv.graphics.TVContainer` defined in Java TV [53] shall be supported.

11.4.1.3 Extended graphics

See U, "(normative): Extended graphics APIs" on page 493.

The class `org.davic.awt.Color` defined in DAVIC 1.4.1p9 [3] shall be supported.

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. [34] 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 when playing MPEG content delivered in MPEG transport streams is defined as follows:

- when the media stream includes the MPEG-2 / DSM-CC Normal Play Time information, the media time of JMF is directly the value of the Normal Play Time.
- otherwise, the media time of JMF is just a time value that progresses as the media stream is played, but whose actual value is implementation dependent.

For a JMF player which is presenting a DVB service, the following rules will be followed in the order given to decide which elementary streams will be presented when multiple audio and video elementary streams are present in a service:

a) Any signalling in the network used to specify default stream selection will be used.

b) For audio and subtitle streams in different languages, user preferences will be used to determine which streams are selected.

c) The streams which are first in the network signalling information (i.e. in the appropriate SI table) will be presented.

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.

11.4.2.3 Extensions to the Framework

11.4.2.3.1 DVB specified extensions

The classes and interfaces defined in annex N, "(normative): Streamed Media API Extensions" on page 340 of this specification are included.

11.4.2.3.2 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].

- MediaTimeEventControl
- MediaTimeEvent
- MediaTimeEventListener
- StreamEventControl
- StreamEvent
- StreamEventListener
- InvalidEventIDException
- MediaPresentedEvent
- MediaLocator
- MediaTimePositionControl
- FreezeControl
- ResourceWithdrawnEvent
- ResourceReturnedEvent
- MediaTimePositionChangedEvent
- LanguageNotAvailableException
- NotAuthorizedException
- NotAuthorizedMediaException
- MediaFreezeException
- InvalidEventNameException

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 `setMediaTime()` on the `MediaTimePositionControl` will be signalled by a `org.davic.media.MediaTimePosition-ChangedEvent`.

11.4.2.3.3 Extensions in javax.tv

The following control will be included from the javax.tv.media package in Java TV [53]:

- MediaSelectControl (the supporting listener and events are also included)
- AWTVideoSizeControl (and AWTVideoSize)

11.4.2.3.4 Required controls for broadcast profiles

The following controls are supported for the broadcast streaming formats specified in 7.2, "Broadcast streaming formats" on page 43:

- org.davic.media.LanguageControl
- org.davic.media.AudioLanguageControl
- org.davic.media.SubtitlingLanguageControl
- org.davic.media.FreezeControl
- javax.tv.media.MediaSelectControl
- org.davic.media.MediaTimePositionControl (for audio from memory only)
- javax.tv.media.AWTVideoSizeControl
- org.dvb.media.VideoPresentationControl
- org.dvb.media.BackgroundVideoPresentationControl
- org.dvb.media.SubtitlingEventControl
- org.dvb.media.VideoFormatControl

11.4.2.3.5 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 157.

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 `MediaSelectSucceed-edEvent` 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.

11.4.2.4 Restrictions on the Framework for Broadcast

Controls that are supported the MHP need not have an associated GUI component. Any calls to `Control.getCon-trolComponent()` 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 shell 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.

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.

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 384.

Relative file names used to access objects in the carousel shall be taken as being relative to the base directory indicated in 10.9.1, "DVB-J application descriptor" on page 81. Calling `new DSMCCObject(".")` or `new java.io.File(".")` will instantiate the directory object that refers to the base directory as indicated in 10.9.1, "DVB-J application descriptor" on page 81.

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 has been run, the directory entry information in the object may be in unloaded state. However, it may also be loaded if the implementation has the information in cache.
- if the directory entry is in unloaded state, it shall be synchronously loaded when any of the following methods are called: canRead(), exists(), isDirectory(), isFile(). If the loading fails, all these methods shall return false.
- after the constructor has been run, the content of the object may be in unloaded state. However, it may also be loaded if the implementation has the information in cache.
- if the content is in unloaded state, it shall be synchronously loaded when the list() method is called for a directory. If this implicit load should result in a service transfer, it shall not be done implicitly and the list() shall return an empty list.
- the method lastModified() returns moduleVersion from the DII for the module that carries the file (treating the octet as an unsigned integer).
- any version changes in a file after the constructor for an InputStream is called will not be visible in the data read from that InputStream.

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()`,`mkdirs()`, `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(), mkdirs() and renameTo() return false to indicate that the request failed

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.

a) The javax.tv.net package as defined in Java TV [53] is included.

b) 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)
```

c) 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

e) The class `org.dvb.net.DatagramSocketBufferControl` shall be supported see Q, "(normative): Datagram Socket Buffer Control" on page 431.

f) Behaviour if unsupported:

When the application tries to joinGroup() on a Multicast address, ProtocolException shall be thrown to indicate failure on joining the group.

11.5.3 Support for IP over the Return Channel

Where support for IP over the return channel is included, all of the `java.net` package shall be included. Platforms not implementing specific optional return channel protocols shall fail as defined in the specification of this API.

On devices whose return channel can be connected or disconnected, connecting a `java.net.Socket` or a `java.net.URLConnection` to a host addressed via the return channel shall automatically setup a connection to the default connection target subject to the application having return channel permission for the default ISP. Such connections shall be automatically disconnected after a time out period defined in the Navigator.

See 11.10, "Java permissions" on page 103.

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 434.

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 242.

The "persistent.root" property which can be obtained from java.lang.System.getProperty() identifies the directory at the root of the file name space used for persistent storage. This value shall be the same for all applications. Accessing any files or directories in the parent directory of this root or directories above that shall throw a SecurityException as defined in the specification for the java.io package. Applications shall have read only access to this root directory. Applications shall have automatic read and write access to the sub-directories <organisation_id> and <organisation_id>/<application_id>. 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 section 11.8.1 of this specification. These directories must be automatically created by the platform. The owner of the organisation directory shall be the platform and it shall always have organisation read / write and world read access. The platform shall set the owner of the application directory to the application with read and write access for that application and no access for other applications.

All files in persistent storage shall store the 48 bit application identifier of their creator as the "owner" of the file. Only the owner of the file is entitled to change the file attributes and file access rights. The owner of a file cannot be changed once a file is created. Applications may only create files or directories in directories to which they have write access. The existing semantics for the java.io package are respected.

11.6 Service Information and Selection APIs

11.6.1 DVB Service Information API

The DVB specific SI API is defined in annex M, "(normative): SI Access API" on page 262.

11.6.2 Service Selection API

The service selection API is defined by the `javax.tv.service.selection` package from Java TV [53].

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 presenting on the background video device.

Applications shall re-acquire the list of service content handlers after a service selection completes. It is implementation dependent whether any previous handlers are re-used. JMF players which are not re-used are stopped and may be disposed by the platform.

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 may throw `java.lang.SecurityException`:

- `NetworkInterfaceController.reserve`
- `NetworkInterfaceController.reserveFor`

See also 11.8.3, "Additional permissions classes" on page 101.

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`

See also 11.8.3, "Additional permissions classes" on page 101.

11.6.5 Protocol Independent SI API

The protocol independent SI API is defined by the following packages from Java TV [53]:

- `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 377.

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 [53].

11.7.1.1 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 88.
- 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.

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 456.

11.7.3 Inter-Application Communication API

This API is formed of the java.rmi package as specified in JavaRMI [35].

To avoid application name collisions, the application that exports the Remote interface shall adopt a name space convention to identify those objects with the RMI registry. The application shall encode a hexidecimal <organisation_id> string and a hexadecimal <application_id> string. These strings correspond to fields defined in 10.5, "Application identification" on page 68.

The application shall prepend the characters "//" before the <ProviderId> string and place the character "/" between the <organisation_id> string and <application_id> string. It shall add a trailing "/" character on the end of the string. An example of the schema is:

platform://01234567/89AB/

NOTE: The name space for applications bound to a single platform is flat. The presence of the "/" character does not mean that the platform supports a name graph that applications traverse. The purpose of the notation is just to signal that the scope is a single platform and to avoid name collisions.

The methods in this API shall not throw SecurityExceptions, regardless of whether usage of java.rmi for inter-application communication will be inside or outside the default sandbox. The potentially sensitive Naming.list()/lookup() methods shall only return those remote objects with which the application may communicate. If the security policy dictates that an application may not use java.rmi for inter-application communication, then the Naming.list() method shall return an empty list and each lookup shall simply fail.

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`.

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 may 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.

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 [53].

The signature of the `org.davic.net.Locator` class will be extended with:

```
"implements javax.tv.locator.Locator"
```

The `create` 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" on page 161).

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].

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 [38] are supported.

11.8.1.1 java.security

From the `java.security` package:

- `AccessControlContext`
- `AccessControlException`
- `AccessController`
- `AllPermission`
- `BasicPermission`
- `CodeSource`
- `GeneralSecurityException`
- `Guard`
- `Permission`
- `PermissionCollection`
- `Permissions`
- `Policy`
- `PrivilegedAction`
- `PrivilegedActionException`
- `PrivilegedExceptionAction`
- `ProtectionDomain`
- `SecurityPermission`
- `UnresolvedPermission`
- `UnresolvedPermissionCollection`

11.8.1.2 java.security.cert

- `Certificate`
- `CertificateEncodingException`
- `CertificateException`

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.net.NetPermission
- 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 [62]:

- `javax.net`
- `javax.net.ssl`
- `javax.security.cert`

The following classes from the `java.security` package shall be supported:

- `InvalidKeyException`
- `Key`
- `KeyException`
- `NoSuchAlgorithmException`
- `NoSuchProviderException`
- `Principal`
- `PublicKey`
- `SignatureException`

Also see 12.10, "Security on the return channel" on page 132.

11.8.3 Additional permissions classes

See T, "(normative): Permissions" on page 487.

11.9 Other APIs

11.9.1 Timer Support

This API is formed of the Timer API defined in Java TV [53] 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]

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 38 shall be included in the property set of the `java.lang.System` class. Thus these properties can be retrieved using `java.lang.System.getProperty()`.

Table 38 : System properties for profile and version interrogation

Property	Semantics	Possible values	Example
mhp.profile.enhanced_broadcast	Indicates whether the enhanced broadcast profile is supported	"YES", "NO"	"YES"
mhp.profile.interactive_broadcast	Indicates whether the interactive broadcast profile is supported	"YES", "NO"	"NO"
mhp.profile.internet_access	Indicates whether the internet access profile is supported	"YES", "NO"	"NO"
mhp.eb.version.major	Major version number of the supported enhanced broadcast profile	Non-negative integer value	"1"
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"

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 39 lists the currently defined options.

Table 39 : System properties for optional feature interrogation

Option	Property
IP Multicast over Broadcast Channel	mhp.option.ip.multicast

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 should 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 the following properties: file.separator, path.separator, line.separator.

11.10.1.4 java.lang.RuntimePermission

This permission should be denied for both unsigned and signed applications.

11.10.1.5 java.io.SerializablePermission

This permission should be denied for both unsigned and signed applications.

11.10.1.6 java.io.FilePermission

A read permission should 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

The MHP has not put any constraints on the usage of the ServiceContext objects. All applications shall be granted a javax.tv.service.selection.ServiceContextPermission with a name string "getServiceContentHandlers" and action string "own".

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 115). 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 the following propertied: file.separator, path.separator, line.separator, persistent.root.

11.10.2.2 java.io.FilePermission

A read permission should be granted for the subtree under which the implementation mounts the object carousels.

When the permission request file requests the permission to access persistent storage and this is granted, a `FilePermission` that permits access to the persistent storage directory subtree is created.

When there is a persistent storage credential in the permission request file and this is granted, `FilePermissions` are created as follows:

- file path = value of persistent.root property + filename from the credential
- action = string containing "read" and/or "write" as indicated in the credential

11.10.2.3 org.dvb.net.ca.CAPermission

When the permission request file requests the permission to communicate with a CA system and this is granted, a `CAPermission` is created as follows:

The CA system ID string from the permission request file is directly used as the first part of the string used for the `CApermission` constructor, this is concatenated with a colon character and the list of the attribute strings based on the attributes listed as true in the permission request file.

11.10.2.4 org.dvb.application.AppsControlPermission

When the permission request file requests the permission to have additional permissions to control the lifecycle of applications and this is granted, an `AppsControlPermission` is created.

11.10.2.5 org.dvb.net.rc.RCPermission

When the permission request file requests the permission to communicate through the return channel and this is granted, an `RCPermission` is created as follows:

- for the default ISP item, the `RCPermission` is created with "target:default" string.
- for items with phone numbers in them, the string is "target:" + the phone number prefix in the permission request file + "*"

On `org.dvb.net.rc.ConnectionRCInterface`, the method `getCurrentTarget` shall always throw a `SecurityException`. The method `setTarget` shall throw a security exception where the application doesn't have the permission to use the target specified. The method `setTargetToDefault` shall throw a security exception where the application doesn't have either "target:default" or "target:*" permissions.

11.10.2.6 org.dvb.net.tuning.TunerPermission

When the permission request file requests the permission to access the Tuning API and this is granted, an `TunerPermission` is created.

11.10.2.7 javax.tv.service.select.SelectPermission

When the permission request file requests the permission to perform service selection and this is granted, an `SelectPermission` is created with a locator "*" and action string "own" and a `ServiceContextPermission` is created with a name "*" and action "own".

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.

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.

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 115 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.

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 108.

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 'sub-tree'.

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/sub-tree to be verified. The certificate itself is signed by a higher certification authority.

To correctly authenticate a sub-tree there must be a valid "chain" of certificates from the signature to a root certificate as is illustrated in figure 13.

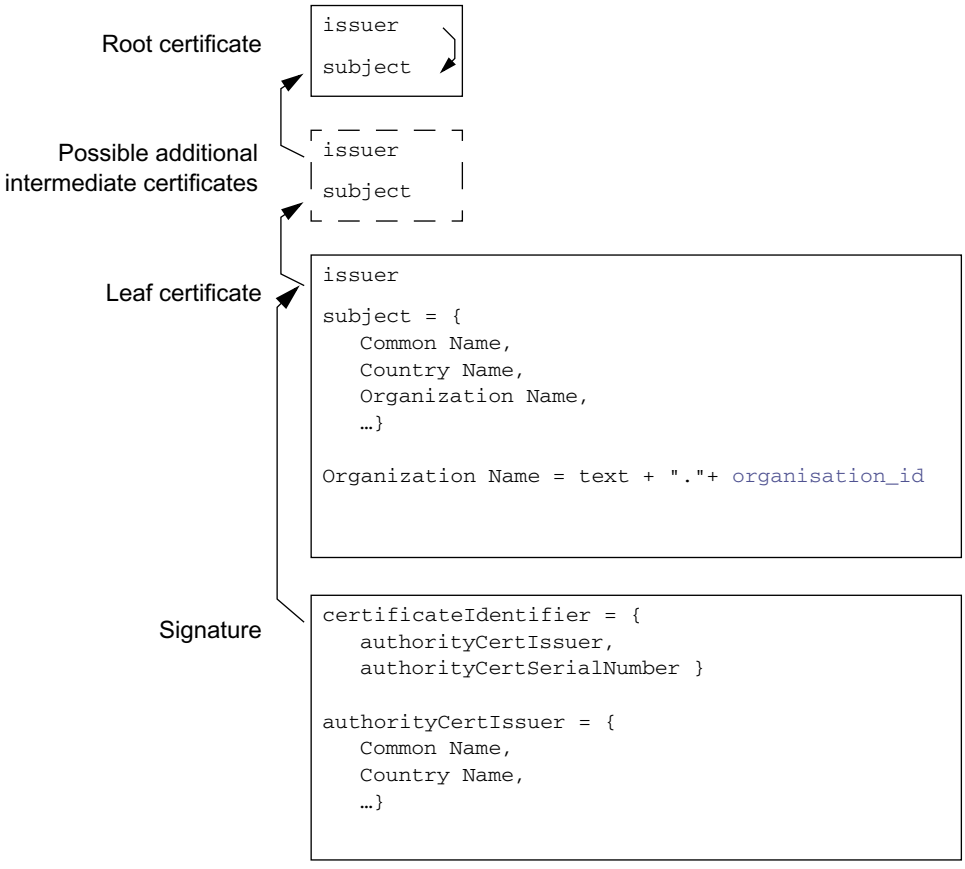


Figure 13 : Certificate chain illustrated

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 sub-trees 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 sub-tree.

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 objects of the carousel are authenticated. For example code files might be authenticated and asset files might be left without authentication.

Finally, this framework embraces key distribution by specifying a certificate mechanism, the aim of this is to certify that the key used to compute the signature is valid and used by a certified service provider. See 12.8, "Procedures for application certificates and signatures" on page 127.

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 108
- "SignatureFile" on page 110
- "CertificateFile" on page 111

These are placed in files in the file system. The location of the file depends on its function.

12.4.1 HashFile

12.4.1.1 Description

The HashFile lists all of the elements of the current directory except itself and the possible signature files. Those elements to be authenticated are associated with hashcodes. The syntax of the HashFile is shown in table 40.

Table 40 : Syntax of the Hashfile (Sheet 1 of 2)

Syntax	Num. Bits	Format
Hashfile () {		
digest_count	16	uimsbf
for(i=0; i<digest_count; i++) {	16	uimsbf
digest_type	8	uimsbf
name_count	16	uimsbf
for(i=0; i<name_count; i++) {		
name_length	8	uimsbf
for(j=0; j<name_length; j++) {		
name_byte	8	bslbf
}		
}		
for(j=0; j<digest_length; j++) {		
digest_byte	8	bslbf
}		
}		

Table 40 : Syntax of the Hashfile (Sheet 2 of 2)

Syntax	Num. Bits	Format
}		
Other data may follow but can be ignored by implementations conforming to this profile.		

digest_count: This 16 bit value identifies the number of digest values in this hash file.

digest_type: This 8 bit value identifies the digest algorithm, if any, used for the associated objects. Table 41 lists the allowed values for this field.

Table 41 : Values of digest_type

value	digest len.	algorithm
0	0	Non authenticated
1	16	MD-5 as defined in RFC 1321 [40]
2	20	SHA-1 as defined in FIPS-180-1 [64]
Other values		Reserved

name_count: This 16 bit value identifies the number of object names associated with the digest value.

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 41.

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 109.

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 application authentication" on page 124.

12.4.1.3 Digest value computation rules

The digest value is computed over the objects named in the HashFile in the order listed. The length of the list of objects associated with each digest value may be one or more.

Each list of objects may contain an arbitrary mix of different object types (e.g. a mixture of file and directory names). The digest value is computed by first initialising the digest algorithm in an algorithm specific way and then applying the relevant data for each object to the algorithm in order. The relevant data for each object depends on its type and is specified in table 42.

Table 42 : Data required for digest value computation

Object type	Relevant data
File	The entire content of the file
Directory	The content of the HashFile of the named directory

12.4.1.4 Special authentication rules

a) For objects which are directories, if the digest_type is non-zero there shall be a HashFile in the sub-directory listed. If the HashFile is absent then the authentication fails.

b) Each HashFile shall provide a complete list of all the objects named in the directory. The authentication shall fail if the set of objects listed in the HashFile is different to the set of objects in the directory. This applies regardless of the value of digest_type associated with the object.

12.4.2 SignatureFile

12.4.2.1 Description

The SignatureFile is a File containing one digital signature. It contains the following ASN.1 DER structure:

```
Signature  ::=  SEQUENCE  {
   certificateIdentifier                    AuthorityKeyIdentifier,
   hashSignatureAlgorithm                   HashAlgorithmIdentifier,
   signatureValue                           BIT STRING  }
```

certificateIdentifier : As defined in the ITU-T X.509 [56] 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 ::=
   {joint-iso-ccitt(2) country(16) us(840) gov(101) csor(3) pki(4)
   algorithm(8) hash(2) sha(3191)}
```

signatureValue: See "signatureValue" on page 135.

12.4.2.2 SignatureFile location and naming conventions

The SignatureFile is located in the root directory of the subtree that contains the application. There can be several SignatureFiles, as there can be several entities that sign the structure.

By convention, the name of a SignatureFile is:

```
'dvb.signaturefile.x'
```

where "x" is a string that distinguishes the possibly several signature files.

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 109.

12.4.3 CertificateFile

12.4.3.1 Description

The CertificateFile contains all of the certificates in the certificate chain in order with the leaf certificate placed first in the file. The encoding of the certificate is defined in ITU-T X.509 [56]. Below is defined the profile of ITU-T X.509 [56] for use in authenticating MHP applications. This profile is based on RFC 2459 [60].

The syntax of the CertificateFile is shown in table 43.

Table 43 : Syntax of the CertificateFile

Syntax	Num. Bits	Format
`Certificatefile () {`		
`  certificate_count`	16	uimsbf
`  for( i=0; i<certificate_count; i++ ) {`		
`    certificate_length`	24	uimsbf
`    certificate()`		
`  }`		
`}`		

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 [56]. See 12.11.1, "Main part of the certificate" on page 135.

12.4.3.2 ASN.1 encoding

The basic specification of the ASN.1 DER encoding used in RFC 2459 [60] is given in ASN.1 [59]. However, RFC 2459 [60] 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 112.

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 107, a key can be authenticated through a 'certificate chain'.

A certificate chain is a hierarchy of certificates that enable the implementation to verify the validity of the key used to check a signature. In the MHP environment, the root certificate is embedded in the MHP. Hence, the file structure shall carry all of the certificate chain apart from the root certificate.

The certificate chain that leads to the public key of one signature shall be placed in the same directory as the signature file.

The name of a CertificateFile is:

```
'dvb.certificates.x'
```

where "x" is a string that is identical to the corresponding portion of the file name of the signature file authenticated by this certificate chain file. See 12.4.2.2, "SignatureFile location and naming conventions" on page 110.

12.5 Profile of X.509 certificates for authentication of applications

This section identifies how ITU-T X.509 [56] 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 [60]. This section identifies the differences from the profile in RFC 2459 [60]. Section 12.11, "The internet profile of X.509 (informative)" on page 134 summarises the profile in RFC 2459 [60].

12.5.1 signatureAlgorithm

This specification supports 2 signature algorithms: MD5 with RSA and SHA with RSA.

12.5.1.1 MD5 with RSA

The signature algorithm with MD5 and the RSA encryption algorithm is defined in RFC 2313 [58]. As defined in RFC 2313 [58], 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 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 [58]. 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 [58].

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 114.

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 112)
- The minimum set of attributes that shall be present in the subject are:

  - `commonName`

  - `countryName`
- If the certificate is a "leaf certificate" (see figure 13, "Certificate chain illustrated" on page 107) 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
  ```

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 218.

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 [58].

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 44 : 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.	May be present and possibly marked critical when both the digitalSignature and keyCertSign bits are set. If these bits are not set then the certificate shall be ignored by implementations.
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 45 : Key for table 44

Keyword	When applied to …	
	broadcasts	receivers
Mandatory	Certificates shall include these extensions.	Receievers shall observe the semantic for this information when provided.
Optional	Certificates may or may be not include these extensions. The MHP specification does not define the use of these extensions and hence broadcasters should not use them.	Receivers can ignore these extensions if present.
Critical / not-critical	Broadcasters should not mark the receiver optional extensions as critical.	Certificates containing unrecognised critical extensions shall be considered as invalid. Receivers should recognise all the extensions that can be critical.

12.6 Security policy for applications

12.6.1 General principles

This section specifies the resource access policy for the downloaded applications. The resource access policy depends on two factors

- The access rights requested by the broadcaster through the signalling
- The access rights granted by the user.

The ultimate access rights that are granted to the applications are the intersection of the access rights requested by the broadcaster and the access rights granted by the user.

Unsigned applications have limited access to platform resources.

By default, 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 persistant 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.

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 may throw a security exception. 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 100).

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 (ie because its actual hash value does not match the expected hash value), then the class shall be considered as not available.

When an application wants to access 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 an IOException shall be raised by the appropriate methods.

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.

```
<!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"
>

<!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)>
```

12.6.2.2 Example

```
<?xml version="1.0"?>
<!DOCTYPE fontdirectory PUBLIC "-//DVB//DTD MHP
permissionrequestfile//EN"
"http://www.dvb.org/DTD/MHP/permissionrequestfile.xml">

<permissionrequestfile orgid="0x000023d2" appid="0x0020">

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

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

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

   <returnchannel>
      <defaultisp></defaultisp>
      <phonenumber>+3583111111</phonenumber>
      <phonenumber>+3583111112</phonenumber>
      <phonenumber></phonenumber>
   </returnchannel>

   <tuning value="false"></tuning>

   <servicesel value="true"></servicesel>

   <userpreferences read="true" write="false"></userpreferences>

   <network>
      <host action="connect">hostname</host>
   </network>

   <persistentfilecredential>
      <grantoridentifier id="0x0202030"></grantoridentifier>
      <expirationdate date="24/12/2032"></expirationdate>
      <filename read="true" write="false">5/15/dir1/scores</filename>
      <filename read="true" write="false">5/15/dir1/names</filename>
      <signature>02320329329293293292149314392942394329423943294239432
      </signature>
      <certchainfileid>3</certchainfileid>
   </persistentfilecredential>

</permissionrequestfile>
```

12.6.2.3 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 is depends on the application type as is shown in table 46 for the types defined in this specification.

Table 46 : Application name for different application types

Application type		File identified for application name
Value	Meaning	
0x0001	DVB-J	The name initial_class_byte, see table 32 on page 82
0x0002	DVB-HTML	The name initial_path_bytes, see table 34 on page 83

This file shall be located in the same directory as the initial file.

12.6.2.4 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 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 47 :

Elements	Comments
`grantoridentifier`	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: `"0x" 0*8hex ; e.g. 0x34243`
`expirationdate`	This element contains the expiration date of this credential.The implementaion should ignore the certificate if the date has expired. The `CDATA` attribute `date` shall follow the following BNF syntax: `2dec "/" 2dec "/" 4dec ; e.g. "01/12/2001"`

Table 47 :

Elements	Comments
`filename`	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: `1*(alphanum | "/" | "*" | "." | "-" | "_" | "?")` "*" 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 The file path shall not start with a "/". It is relative to the path obtained from the persistent.root property.. The element has two attributes read and write that can take the value "true" or "false".
`signature`	This element contains a signature from the grantor. Signature structure is as defined section 13.3.2.1 of TAM232r12. The siganture follows the following BNF syntax: `1*dec`
`certchainfileid`	This element contains the identifier of the certificate chain, i. e. the "x" in the file name "dvb.certificate.x".

The signature is computed on the binary concatenation of the following fields on the following order:

Table 48 :

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.:
 	
for (i=0;i<filenumber;i++) {
read
write
filepath
}
	
	4 or 5 char ("true" or "false")
	4 or 5 char ("true" or "false")
	string in ASCII

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 111. Certification chain authenticates grantor's certificate. This chain shall derive from one of the root authorities embedded in the MHP.

12.6.2.5 File Access

12.6.2.5.1 Unsigned applications

Have no access to the persistent storage

12.6.2.5.2 Default policy for signed applications

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

When an application is granted access to the File system, the file access policy is derived from the policy used in the Unix world:

An Application owns the files it has created. The file owned by an application identified by OID/AppId has to be located in a subdirectory of /home/OID/AppId or directly in /home/OID/AppID. By default, it cannot access any file outside the /home/OID/AppID.

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.

See org.dvb.io.persistent see K, "(normative): DVB-J persistent storage API" on page 242.

12.6.2.5.3 Permission request syntax

```
<!ELEMENT file EMPTY>
<!ATTLIST file
 value (true|false) "true"
>
```

12.6.2.6 CA API

12.6.2.6.1 Unsigned applications

An unsigned application cannot access the following methods : `CAModule.buyEntitlements`, `CAModule.openMessageSession`, `CAModuleManager.addMMIListener`, `CAModule.queryEntitltements`, `CAModule.listEntitlements`.

12.6.2.6.2 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 : `buyEntitlements`, `openMessageSession`, `sendToModule` (in the `CAModule` class).

The permission request file requests the MHP to grant additional rights to the application with the ConditionalAccess Permission described below.

12.6.2.6.3 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"
>
```

The string specifying the CA system IDs has the following syntax:

```
CAIds          =  1*CASystemId | "[" CASystemId "-" CASystemId "]" | "*"
CASystemId     = "0x" 4*hex
hex            = digit | "A" | "B" | "C" | "D" | "E" | "F" | "a" | "b" |
                   "c" | "d" | "e" | "f"
digit          = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

12.6.2.7 Application lifecycle control policy

Applications shall not launch broadcast applications that are not in the same service.

12.6.2.7.1 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.

12.6.2.7.2 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.

12.6.2.7.3 Syntax

```
<!ELEMENT applifecyclecontrol EMPTY>
<!ATTLIST applifecyclecontrol
 value (true|false) "true"
>
```

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.

12.6.2.8 Return channel access policy

12.6.2.8.1 Unsigned applications

An unsigned application may not use the return channel.

12.6.2.8.2 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.

12.6.2.8.3 Return channel permission syntax

```
<!ELEMENT returnchannel (defaultisp?,phonenumber+)>
<!ELEMENT phonenumber (#PCDATA)>
<!ELEMENT defaultisp EMPTY>
```

The syntax of the phone number string is as defined below.

```
phone_number= "+" 1*digit | 1*digit
digit    = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

If default ISP is not present, then the net API shall not use the default ISPnumber.

The presence of the `defaultisp` tag indicates that the application is allowed to use the default ISP connection. If this tag is not present, the application is not allowed to use the default ISP connection.

When a phone number is given, this number defines a prefix of the allowed phone numbers and applications are allowed to call to all numbers that start with one of the prefixes defined in the permission request file, subject to the MHP terminal granting this right.

NOTE: By defining an empty phone number tag (i.e. empty string as the prefix), the application could try to call to any phone number.

12.6.2.9 Tuning access policy

12.6.2.9.1 Unsigned applications

An unsigned application may not tune using the Tuning API.

12.6.2.9.2 Signed applications

By default, an 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.

12.6.2.9.3 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.

12.6.2.10 Service selection policy

12.6.2.10.1 Unsigned applications

Unsigned applications may not select a new service.

12.6.2.10.2 Signed applications

By default, signed applications can select any new service, unless otherwise specified in the permission request file.

12.6.2.10.3 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.

12.6.2.11 Media API access policy

The media API is inside the sandbox.

12.6.2.12 Inter-application communication policy

12.6.2.12.1 Unsigned applications

Unsigned applications may communicate with each other through the inter-application communication API. However, an unsigned application may NOT communicate with a signed application through the inter-application communication API

12.6.2.12.2 Signed applications

Signed applications signalled in the same service may communicate with each other through the inter-application communication API.

No special Permission need be defined.

12.6.2.13 User Setting and Preferences access policy

12.6.2.13.1 Unsigned applications

Read access to User Language, Morality Level, FontSize, Country Code.

An unsigned application cannot access (neither read nor write) to other preferences

12.6.2.13.2 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.

12.6.2.13.3 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.

12.6.2.14 Network permissions

12.6.2.14.1 Unsigned applications

Unsigned applications do not have access to the return channel and therefore do can not access remote network hosts,

12.6.2.14.2 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.

12.6.2.14.3 Permission syntax

```
<!ELEMENT network (host)+>
<!ELEMENT host (#PCDATA)>
<!ATTLIST host
 action CDATA #REQUIRED
>
```

The two strings that specific the host and the action shall be formatted as specified in the `java.net.SocketPermission` class as defined in 11.8, "Security" on page 100.

12.6.2.15 Dripfeed permissions

12.6.2.15.1 Unsigned applications

Has no access to drip feed mode.

12.6.2.15.2 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`.

12.6.2.15.3 Permission request syntax

```
<!ELEMENT dripfeed EMPTY>
<!ATTLIST dripfeed
   value (true|false) "true"
>
```

12.7 Example of application authentication

12.7.1 Scenario Example

This section is informative and gives an example of how an Object Carousel carrying two applications can be organized.

In this example, the Object Carousel carries two signed applications: Xlet1 and Xlet2.

The main class of Xlet1 is found at root/Xlet1/classes/Xlet1.class and this application further comprises the following classes:

- root/Xlet1/classes/foo1.class
- root/Xlet1/classes/subclasses/sub1.class
- root/Xlet1/classes/subclasses/sub2.class

Xlet1 consumes data located in root/Xlet1/data/Xlet1.dat. This file is not authenticated.

The main class of Xlet2 is found at root/Xlet2/classes/Xlet2.class and this application further comprises the following classes:

- root/Xlet2/classes/foo2.class,

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 digest algorithm used is MD5 only. The two applications are signed with the same private key. The corresponding public key is supposed to be found in the X.509 certificate, with serial number 0123456.

The file structure is shown below :

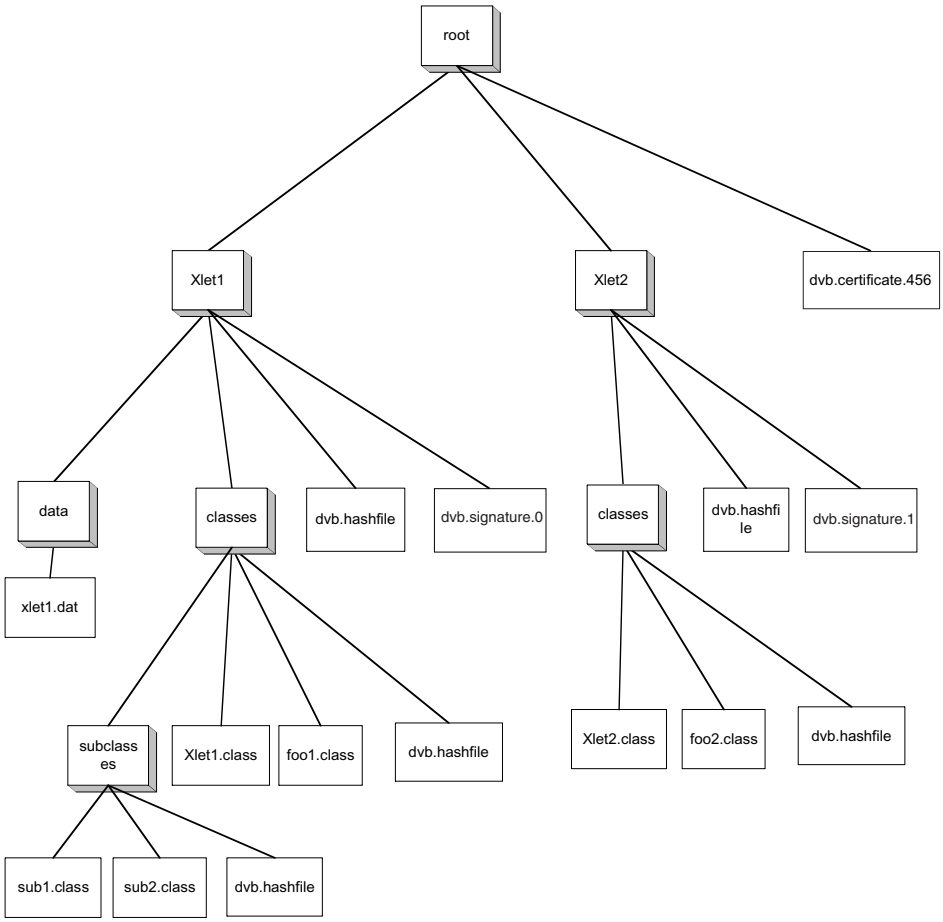


Figure 14 : Example of an authenticated file tree

12.7.2 Hashes and signature computations:

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

a) compute the hash of root/Xlet1/classes/subclasses/sub1.class -> H0

b) compute the hash of root/Xlet1/classes/subclasses/sub2.class -> H1

c) create the file dvb.hashfile. Its content is as follows:

Table 12-1:

Field	Comment
1	One type of digest algorithm
1	First type of digest algorithm = MD5
2	Number of entries of the directory for which a MD5 hash has been computed
Sub1.class	
H0	MD5 hash of file sub1.class
Sub2.class	
H1	MD5 hash of file sub1.class

d) put this hashfile under the of root/Xlet1/classes/subclasses directory

e) compute the MD5 hash of dvb.hashfile -> H2 (H2 becomes the hash value of the of root/Xlet1/classes/subclasses directory)

12.7.2.2 Computation of the hashes of the of root/Xlet1/classes directory

f) compute the MD5 hash of root/Xlet1/classes/Xlet1.class -> H3

g) compute the MD5 hash of root/Xlet1/classes/foo1.class -> H4

h) create the file dvb.hashfile. Its content is as follows:

Table 12-1:

Field	Comment
1	One type of digest algorithm
1	First type of digest algorithm = MD5
3	Number of entries of the directory for which a MD5 hash has been computed
Subclasses	Subclasses directory
H2	MD5 hash of subclasses directory
Xlet1.class	
H3	MD5 hash of file Xlet1.class
Foo1.class	
H4	

i) put this dvb.hashfile under the root/Xlet1/classes directory

j) compute the has of this dvb.hashfile -> H5 (which becomes the hash value of this directory)

12.7.2.3 Computation of the hashes of the of root/Xlet1 directory

k) the root/Xlet1/classes entry is the only authenticated entry of the root/Xlet1 directory. So the dvb.hashfile contained in root/Xlet1 can be computed already. It is as follows:

Table 12-1:

Field	Comment
2	two types of digest algorithm
1	First type of digest algorithm = MD5
1	Number of entries of the directory for which a MD5 hash has been computed
Classes	
H5	MD5 hash of the classes directory
0	Second type of digest algorithm (0 => non authenticated data)
Data	This ‘data’ directory is not authenticated

l) put this dvb.hashfile under root/Xlet1 directory.

12.7.2.4 Computation of the signature

m) from now on, the whole subtree can be signed: compute the hash value of root/Xlet1/dvb.hashfile, and RSA-encrypt the result with the private key corresponding to the public key that can be found in the certificate whose serial number is 0123456.

The following structure has then to be ASN.1 encoded:

- AuthorityCertIssuerName: Name of the CA
- AuthorityCertSerialNumber: 0123456
- HashSignatureAlgorithm: MD5
- SignatureValue: result of step 13.

n) put this structure into the dvb.signature File and put this file under the root/Xlet1 directory.

o) put the dvb.certificate.456 file containing the X.509 certificate under the root directory (It could also have been put in the Xlet1 directory, but since the application Xlet2 is signed with the same key, the better location is of course the root directory).

p) apply the same mechanism for Xlet2 directory and subdirectories.

12.8 Procedures for application certificates and signatures

See 12.12, "MHP certification procedures" on page 142.

12.9 Certificate management

12.9.1 Certificate Revocation Lists

12.9.1.1 Introduction (informative)

Certificates may be revoked prior to their expiration time, e.g. if the broadcaster's private key is assumed to be compromised, or the broadcaster is no longer to be certified by the CA. Each CA publishes a list of revoked certificates, called a CRL (Certificate Revocation List). This contains the list of the serial number of revoked certificates.

During the validation process of a certificate chain, the CRL of each certification authority on the certification path is checked.

12.9.1.2 Distribution of CRLs (informative)

Two routes from the broadcaster to the MHP terminal can be envisaged:

- via the return channel
- in the broadcast MPEG stream

12.9.1.2.1 Distribution via return channel

In the certificate extension fields, there is an optional field called "CRL Distribution points". This field can hold a URL pointing to a crlFile which can be downloaded.

This approach is suitable for obtaining CRLs relating to TLS authentication of return channel exchanges. It is not suitable for delivering CRLs for broadcast application authentication as:

- not all MHP profiles require return channel support
- MHP terminals may be able to receive broadcast applications when not connected to a return channel
- it would not be acceptable to require a return channel session to authenticate each broadcast application

12.9.1.2.2 Distribution via MPEG stream

For an MHP terminal without a working return channel, the only way to deliver CRLs is via the broadcast MPEG Transport Stream.

12.9.1.3 CRL retention

12.9.1.3.1 Requirement

MHP terminals shall retain CRLs in persistent storage because:

- this enhances security as it defends against attacks with signals that filter out the CRL and use a revoked certificate
- it is more efficient to cache CRLs rather than downloading the CRLs for authentication of each application

12.9.1.3.2 Storage requirement

The minimum amount of persistent memory required to store CRLs is addressed in 12.12, "MHP certification procedures" on page 142.

12.9.1.3.3 Storage management

The broadcast CRL and RCMM signalling (see 12.9.2, "Root certificate management" on page 130) manages the use of the persistent storage.

If the CRL of a non-root certificate CA become's too large the CA's certificate itself can be revoked by its parent CA who adds it to their CRL. For root certificates, the RCMM mechanism can be used.

The broadcaster policy in this area is addressed in 12.12, "MHP certification procedures" on page 142.

12.9.1.4 CRL file location and naming convention

The format of the name of files carrying CRLs shall be:

```
'dvb.crl.x'
```

where the `'x'` portion of the filename corresponds to the `'x'` portion of a certificate filename. See 12.4.3.5, "Certificate-File location and naming conventions" on page 112.

CRL files for root certification authorities shall be located in the root directories of the file system. CRL files for non-root certification authorities shall be located in the same directory as certificate issued by that certificate authority.

Typically CRL files are not provided for every certificate, but rather are inserted on a sample basis where they are likely to be visited.

The CRL file shall be accompanied by the certificate that contains the public key which is needed to check the signature of the CRL. This certificate shall be placed in the same directory as the CRL, i.e. the root directory of the file system. The location and name of the file containing the certificate shall follow the same rules as for the certificates used for application authentication, described in 12.4.3.5 on page 112.

12.9.1.5 Examples

12.9.1.5.1 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.

12.9.1.5.2 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).

12.9.1.6 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 [56] 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 134:

```
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
 }
```

12.9.1.7 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 112. This applies to the following fields:

signatureAlgorithm : follows "signatureAlgorithm" on page 112.

signatureValue: follows "signatureValue" on page 113.

version: follows "version" on page 113.

signature: follows "signatureAlgorithm" on page 112.

issuer: follows "issuer" on page 113.

thisUpdate: Publication date of this CRL. Follows the encoding of Time used for validity. See "validity" on page 113.

nextUpdate: Publication date of the next version of the CRL. Follows the encoding of Time used for validity. See "validity" on page 113.

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 113.

crlExtensions: The syntax of the Extensions element is reproduced on "Extensions" on page 138. 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 139.

The following crlExtension shall be supported:

- AuthorityKeyIdentifier as defined in section 12.4.2.1 on page 110.

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

Other crlExtensions are optional.

12.9.1.8 CRL Processing

List of revoked serial numbers shall be kept in persistent storage in set top box.

When an MHP terminal finds a file with the file name format given in 12.4.3.5, "CertificateFile location and naming conventions" on page 112, it shall do the following sequence:

a) Get the field thisUpdate and compare it with the last update for the CRL. If thisUpdate is not after the last update, ignore the CRL.

b) Check the signature of the CRL. If it is invalid, ignore the message.

c) Process the content of the CRL message:
- store the list of serial numbers of revoked certificates in persistent storage.
- update the stored value of thisUpdate with the value from the CRL.

d) If a CA's certificate has been revoked, remove its CRL if it was stored in the MHP terminal.

12.9.2 Root certificate management

12.9.2.1 Introduction

Every compliant MHP terminal will have to maintain a set of X.509 root certificates in persistent storage. These root certificates will be placed in the MHP terminal by its manufacturer during the manufacturing process.

It could be necessary to update the set of root certificates for MHP terminals that are already deployed. Possible reasons that could require such an update include:

- a root certificate becoming compromised
- technical developments (such as the emergence of factorisation algorithms) that require the use of greater key lengths in root certificates to provide adequate security
- retirement of a certificate due to its age

So, It is necessary to have a standard mechanism to update this set.

NOTE: A manufacturer specific mechanism could require a different message for each manufacturer and so would be more expensive in terms of broadcast bandwidth.

The mechanism specified here uses messages called RCMM (Root Certificate Management Messages). These messages contain a set of new root certificates to add and a reference to the root certificates to remove.

12.9.2.2 Security of RCMM

RCMM are authenticated by multiple signatures. An RCMM message will be accepted by an MHP terminal if and only if it has at least N signatures.

The initial value of N and the maximum value of N that MHP terminals will ever be asked to support are specified in 12.12, "MHP certification procedures" on page 142.

NOTE: The initial value of N is expected to be 2

The use of multiple signatures guarantees that the set of root certificates can be updated securely even if one of the root certificates has been compromised.

RCMM can update the number of signatures required for future RCMM using the nextNbOfSignature field.

The RCMM message shall be signed with the key of the certificates to be removed.

12.9.2.3 Format of RCMM

The encoding of RCMM is ASN.1 DER (see ASN.1 [59]):

```
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) Get the number signatures from the RCMM, if this number is lower than the minimum required, ignore the message.

c) 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.

d) Check all the signatures of the RCMM. If any of the signatures is invalid, ignore the message.

e) Process the content of the RCMM message, add and remove root certificates according to the RCMM message.

f) Store in persistent storage the date of thisUpdate as the date of the last update.

g) If a root certificate is removed, the CRLs associated are also removed.

The implementation shall ensure the following:

- The integrity of the persistent storage shall be kept if the power supply fails for any reason during the processing of the RCMM message.

  i.e. If the power is switched off during the processing of the RCMM message, the set of root certificates shall remain as if processing of the RCMM message had not started.

- The minimum amount of memory reserved to store the list of root certificate is specified in 12.12, "MHP certification procedures" on page 142. 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 termninal: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 delivers this RCMM message to the broadcasters to update the MHP terminals.

   (The broadcasting period should be long enough to update almost all of the MHP terminals in the field).

f) RCA will provide RC1 and RC2 to set top box manufacturers as the new list of root certificates to put in set top boxes.

12.10 Security on the return channel

General purpose security for the return channel is provided by the TLS (Transport Layer Security) protocol as described in RFC 2246 [65].

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 1 (see RFC 2246 [65] for definition of the terms) which identifies which methods are required in an MHP.

Table 1 : Profile of cipher suites that implementations are required to support

CipherSuite	Key Exchange	Cipher	Hash	Value (hex)	MHP status
TLS_NULL_WITH_NULL_NULL	NULL	NULL	NULL	00, 00	Required
TLS_RSA_WITH_NULL_MD5	RSA	NULL	MD5	00, 01	Required
TLS_RSA_WITH_NULL_SHA	RSA	NULL	SHA	00, 02	Required
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA	RSA_EXPORT	DES40_CBC	SHA	00, 08	Required
TLS_RSA_WITH_DES_CBC_SHA	RSA	DES_CBC	SHA	00, 09	Required
TLS_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES_EDE_CBC	SHA	00, 0A	Required
DVB_RSA_WITH_3DES_EDE112_SHA	RSA	3DES_EDE112_CBC	SHA	FF, 01	Required

12.10.3 The EDE112 cipher

The DVB_RSA_WITH_3DES_EDE112_SHA cipher suite is introduced by this specification. It is the following triple encryption using 2 different DES keys:

$$C = E_{K1}(D_{K2}(E_{K1}(P)))$$

$$P = D_{K1}(E_{K2}(D_{K1}(C)))$$

This algorithm is a variant of TLS_RSA_WITH_3DES_EDE_CBC_SHA where the keys that are used in first and third stage of the encryption process are identical (i.e. k1 = k3).

12.10.4 Downloading of certificates for TLS

12.10.4.1 Introduction

Before the TLS connection can be established, the MHP has to ensure that the certificate list sent by a server contains at least one trusted certificate. In computer environment, this is simply done by checking the list of certificates against one certificate that is resident in the computer.

In the MHP environment, an 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 101. The process of server authentication involves the checking of the certificate chain sent by the TLS server. As described above, the application may trust a certificate that the MHP does not know about. This certificate needs then to be downloaded by the MHP before the TLS session can be established.

This section specifies how the MHP terminal identifies and manages the TLS certificates that are downloaded along with the application and how verification (or otherwise) is presented to the application).

12.10.4.2 Usage of certificate in TLS

12.10.4.2.1 When certificates are delivered with the application

One or several TLS root certificates can be optionally broadcast along with the application.

When the certificate chain sent by the TLS server is not compatible with any of the TLS root certificates sent with the application an `IOException` will be thrown.

12.10.4.2.1.1 Certificate file naming and location

To facilitate certificate chain checking the name of the certificate file shall be:

```
dvb.tls[.organisation_id[.application_id[.xxx]]]
```

where 'xxx' is an optional string discriminating certificates where necessary.

Location of the TLS certificates is application type dependant. See table 2.

Table 2 : 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 82.
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 83.
0x0003…0xFFFF	

12.10.4.2.1.2 Certificate authentication

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

12.10.4.2.2 When no certificates are provided

When there are no TLS certificates sent with the application then the implementation will allow connection to be established to any server. The application can then use the JSSE API (see 11.8.2 on page 101) 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 [60] 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 112.

NOTE: RFC 2459 [60] section 7.2 lists the signature algorithms supported by that profile.

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 [60] 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 113.

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 113.

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.

See 12.5.1, "signatureAlgorithm" on page 112.

NOTE: RFC 2459 [60] 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 [55] 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 113.

12.11.1.9 validity

The certificate validity period is the time interval during which the CA warrants that it will maintain information about the status of the certificate. The field is represented as a SEQUENCE of two dates: the date on which the certificate validity period begins notBefore) and the date on which the certificate validity period ends (notAfter). Both notBefore and notAfter may be encoded as UTCTime or GeneralizedTime.

CAs conforming to this profile MUST always encode certificate validity dates through the year 2049 as UTCTime; certificate validity dates in 2050 or later MUST be encoded as GeneralizedTime.

```
Validity::= SEQUENCE {
        notBefore       Time,
        notAfter        Time }

Time::= CHOICE {
        utcTime         UTCTime,
        generalTime     GeneralizedTime }
```

12.11.1.9.1 UTCTime

The universal time type, UTCTime, is a standard ASN.1 type intended for representation of dates and time. UTCTime specifies the year through the two low order digits and time is specified to the precision of one minute or one second. UTCTime includes either Z (for Zulu, or Greenwich Mean Time) or a time differential.

For the purposes of this profile, UTCTime values MUST be expressed Greenwich Mean Time (Zulu) and MUST include seconds (i.e., times are YYMMDDHHMMSSZ), even where the number of seconds is zero. Conforming systems MUST interpret the year field (YY) as follows:

- Where YY is greater than or equal to 50, the year shall be interpreted as 19YY; and
- Where YY is less than 50, the year shall be interpreted as 20YY.

12.11.1.9.2 GeneralizedTime

The generalized time type, GeneralizedTime, is a standard ASN.1 type for variable precision representation of time. Optionally, the GeneralizedTime field can include a representation of the time differential between local and Greenwich Mean Time.

For the purposes of this profile, GeneralizedTime values MUST be expressed Greenwich Mean Time (Zulu) and MUST include seconds (i.e., times are YYYYMMDDHHMMSSZ), even where the number of seconds is zero. GeneralizedTime values MUST NOT include fractional seconds.

See 12.5.5, "validity" on page 113.

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 136). 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 [60] 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 [60] 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 [60] 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 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 [60] 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 135) 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 113.

NOTE: RFC 2459 [60] 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 114.

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 [56].

See 12.5.9, "Extensions" on page 114.

12.11.2 Standard certificate extensions

See table 44, "Profile for standard certificate extensions" on page 114.

12.11.2.1 Authority key identifier

This extension is used where an issuer has multiple signing keys. It provides a means of finding the public key that can be used to check the signature of the certificate.

The identification can be based on either the keyIdentifier or the on the pair (authorityCertIssuer, AuthorityCertSerialNumber). It is recommended to use the pair (authorityCertIssuer, AuthorityCertSerialNumber) instead of the keyIdentifier because there is no common agreement on the way to compute a unique KeyIdentifier.

```
AuthorityKeyIdentifier ::= SEQUENCE {
  keyIdentifier             [0] KeyIdentifier           OPTIONAL,
  authorityCertIssuer       [1] GeneralNames            OPTIONAL,
  authorityCertSerialNumber [2] CertificateSerialNumber OPTIONAL }
```

12.11.2.2 Subject key identifier

This extension provides of identifying certificates that contain a particular public key.

```
SubjectKeyIdentifier ::= KeyIdentifier
```

12.11.2.3 Key usage

The key usage extension defines the purpose of the key contained in the certificate.

```
keyUsage ::= BIT STRING {
  digitalSignature      (0),
  nonRepudiation        (1),
  keyEncipherment       (2),
  dataEncipherment      (3),
  keyAgreement          (4),
  keyCertSign           (5),
  cRLSign               (6),
  encipherOnly          (7),
  decipherOnly          (8)
}
```

12.11.2.4 Private key usage period

This field is used to defined a period of validity for the private key which is different from the period of validity of the certificate. This is only meaningful for digital signature keys.

```
privateKeyUsagePeriod EXTENSION ::= {
  SYNTAX        PrivateKeyUsagePeriod
  IDENTIFIED BY id-ce-privateKeyUsagePeriod }

PrivateKeyUsagePeriod ::= SEQUENCE {
  notBefore        [0]  GeneralizedTime OPTIONAL,
  notAfter         [1]  GeneralizedTime OPTIONAL }
  ( WITH COMPONENTS {..., notBefore PRESENT} |
    WITH COMPONENTS   {..., notAfter PRESENT} )
```

12.11.2.5 Certificate policies

This field is used to define a policy defining the purpose for which the certificate may be used.

Applications may have a list of specific policies they will accept, the certificate validation software must be able to compare the policy OIDs found in the certificate to that list.

```
certificatePolicies EXTENSION ::= {
  SYNTAX      CertificatePoliciesSyntax
  IDENTIFIED BY id-ce-certificatePolicies }

CertificatePoliciesSyntax ::= SEQUENCE SIZE (1..MAX) OF PolicyInformation

PolicyInformation ::= SEQUENCE {
  policyIdentifier   CertPolicyId,
  policyQualifiers   SEQUENCE SIZE (1..MAX) OF
                     PolicyQualifierInfo OPTIONAL }

CertPolicyId ::= OBJECT IDENTIFIER

PolicyQualifierInfo ::= SEQUENCE {
  policyQualifierId          CERT-POLICY-QUALIFIER.&id
                             ({SupportedPolicyQualifiers}),
  qualifier                  CERT-POLICY-QUALIFIER.&Qualifier
          ({SupportedPolicyQualifiers}{@policyQualifierId})
          OPTIONAL }

SupportedPolicyQualifiers CERT-POLICY-QUALIFIER ::= { ... }
```

12.11.2.6 Policy mappings

This field is used to define equivalence between policies from different CA ‘s policy Domain.

```
policyMappings EXTENSION ::= {
  SYNTAX      PolicyMappingsSyntax
  IDENTIFIED BY id-ce-policyMappings }

PolicyMappingsSyntax ::= SEQUENCE SIZE (1..MAX) OF SEQUENCE {
  issuerDomainPolicy    CertPolicyId,
  subjectDomainPolicy   CertPolicyId }
```

12.11.2.7 Subject Alternative Name

This field is used to define additional identities for the subject name of the certificate (such as an internet email address).

```
subjectAltName EXTENSION ::= {
  SYNTAX    GeneralNames
  IDENTIFIED BY id-ce-subjectAltName }

GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName

GeneralName ::= CHOICE {
  otherName                 [0]  INSTANCE OF OTHER-NAME,
  rfc822Name                [1]  IA5String,
  dNSName                   [2]  IA5String,
  x400Address               [3]  ORAddress,
  directoryName             [4]  Name,
  ediPartyName              [5]  EDIPartyName,
  uniformResourceIdentifier [6]  IA5String,
  iPAddress                 [7]  OCTET STRING,
  registeredID              [8]  OBJECT IDENTIFIER }
```

12.11.2.8 Issuer Alternative Name

This field is similar to the previous one for the issuer identity.

```
issuerAltName EXTENSION ::= {
  SYNTAX      GeneralNames
  IDENTIFIED BY id-ce-issuerAltName }
```

12.11.2.9 Subject Directory attributes

This field is used to carry optional directory attributes associated with the subject.

```
subjectDirectoryAttributes EXTENSION ::= {
  SYNTAX      AttributesSyntax
  IDENTIFIED BY id-ce-subjectDirectoryAttributes }

AttributesSyntax ::= SEQUENCE SIZE (1..MAX) OF Attribute
```

12.11.2.10 Basic Constraints

This field indicates if the subject of the certificate is a certification authority and how deep a certification path may exist through that path.

This extension shall appear in every CA Certificates. It will be used to prevent unauthorised entities to act as a CA.

```
basicConstraints EXTENSION ::= {
  SYNTAX      BasicConstraintsSyntax
  IDENTIFIED BY id-ce-basicConstraints }

BasicConstraintsSyntax ::= SEQUENCE {
  cA                    BOOLEAN DEFAULT FALSE,
  pathLenConstraint     INTEGER (0..MAX) OPTIONAL }
```

12.11.2.11 Name Constraints

This field indicates a namespace within which all subject names in subsequent certificates in a certification path shall be located.

```
nameConstraints EXTENSION ::= {
  SYNTAX      NameConstraintsSyntax
  IDENTIFIED BY id-ce-nameConstraints }

NameConstraintsSyntax ::= SEQUENCE {
  permittedSubtrees          [0]  GeneralSubtrees OPTIONAL,
  excludedSubtrees           [1]  GeneralSubtrees OPTIONAL }

GeneralSubtrees ::= SEQUENCE SIZE (1..MAX) OF GeneralSubtree

GeneralSubtree ::= SEQUENCE {
  base                GeneralName,
  minimum        [0]  BaseDistance DEFAULT 0,
  maximum        [1]  BaseDistance OPTIONAL }

BaseDistance ::= INTEGER (0..MAX)
```

12.11.2.12 Policy Constraints

This field is only used in CA 's certificate (i.e. not in end entity certificate). It is used to prohibit policy mapping or to require that each certificate in a path contain an acceptable policy identifier.

```
policyConstraints EXTENSION ::= {
  SYNTAX      PolicyConstraintsSyntax
  IDENTIFIED BY id-ce-policyConstraints }

PolicyConstraintsSyntax ::= SEQUENCE {
  requireExplicitPolicy   [0] SkipCerts OPTIONAL,
  inhibitPolicyMapping    [1] SkipCerts OPTIONAL }

SkipCerts ::= INTEGER (0..MAX)
```

12.11.2.13 Extended key usage field

This field is an extensions of the previous field keyUsage. It is used to define more purposes for which the certified public key may be used.

```
extKeyUsage EXTENSION ::= {
  SYNTAX      SEQUENCE SIZE (1..MAX) OF KeyPurposeId
```

```
  IDENTIFIED BY id-ce-extKeyUsage }

KeyPurposeId ::= OBJECT IDENTIFIER
```

12.11.2.14 CRL Distribution points

This field defines how CRL (Certificate revocation list) information can be obtained.

```
cRLDistributionPoints EXTENSION ::= {
  SYNTAX         CRLDistPointsSyntax
  IDENTIFIED BY  id-ce-cRLDistributionPoints }

CRLDistPointsSyntax ::= SEQUENCE SIZE (1..MAX) OF DistributionPoint

DistributionPoint ::= SEQUENCE {
  distributionPoint  [0]  DistributionPointName OPTIONAL,
  reasons            [1]  ReasonFlags OPTIONAL,
  cRLIssuer          [2]  GeneralNames OPTIONAL }

DistributionPointName ::= CHOICE {
  fullName                  [0]  GeneralNames,
  nameRelativeToCRLIssuer   [1]  RelativeDistinguishedName }

ReasonFlags ::= BIT STRING {
  unused                (0),
  keyCompromise         (1),
  cACompromise          (2),
  affiliationChanged    (3),
  superseded            (4),
  cessationOfOperation  (5),
  certificateHold       (6) }
```

12.12 MHP certification procedures

This MHP certification procedures reference is a place holder for a document that will have to be written and agreed with the root CA that describe the administration of certificates.

As defined in section 12.5.6 on page 113, each leaf certificate shall contain the organisationId in the organisation name attribute. In case there are several root certificate entities, the certificate procedure shall ensure that the leaf certificate contain the organisationId defined in ETR 162 [10] which is owned by this organisation.

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.

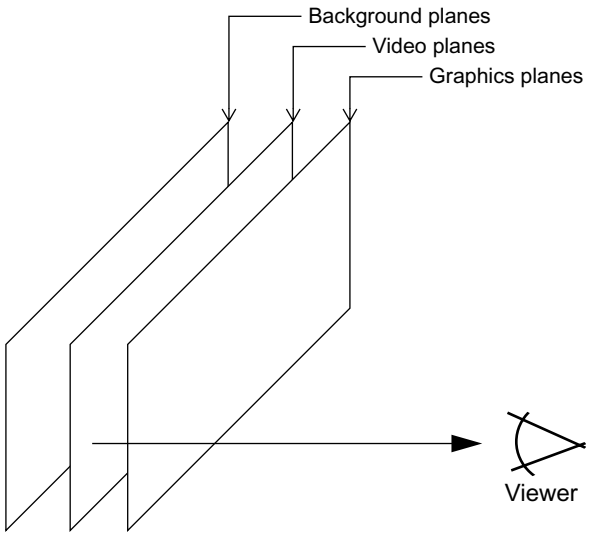


Figure 1 : Illustration of the different types of display planes

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 157.

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 151). 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 218.

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`).

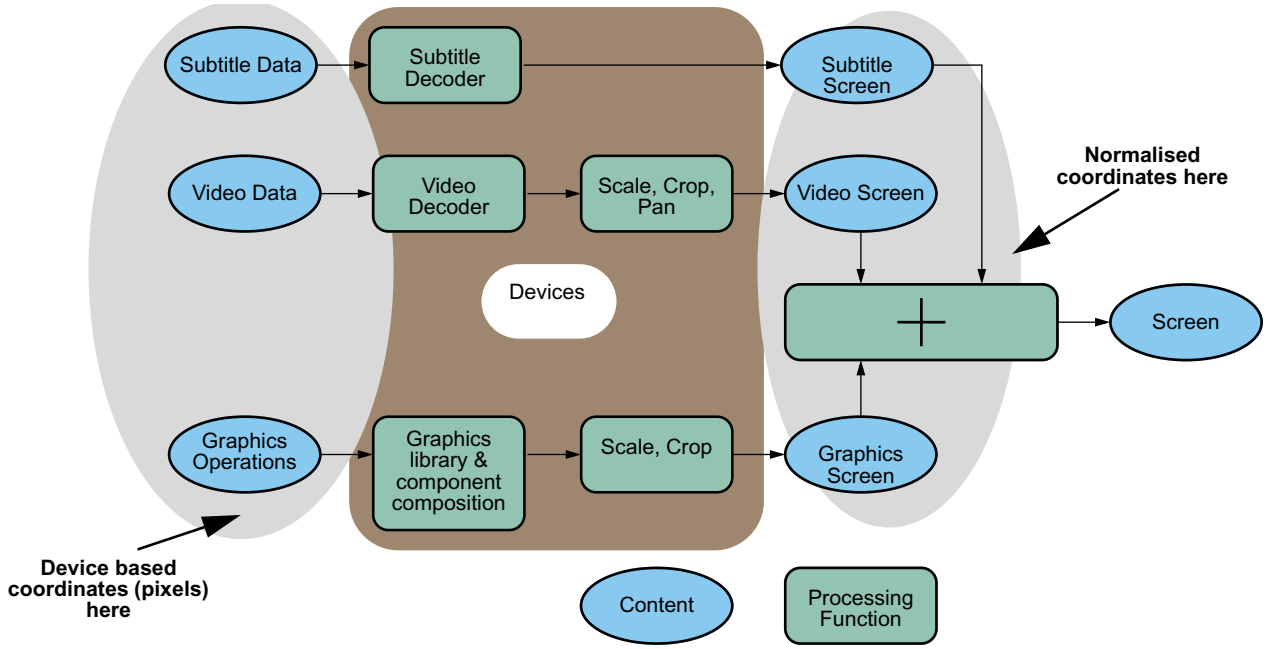


Figure 2 :

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 the 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 `HGraphics-Device`. 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`.

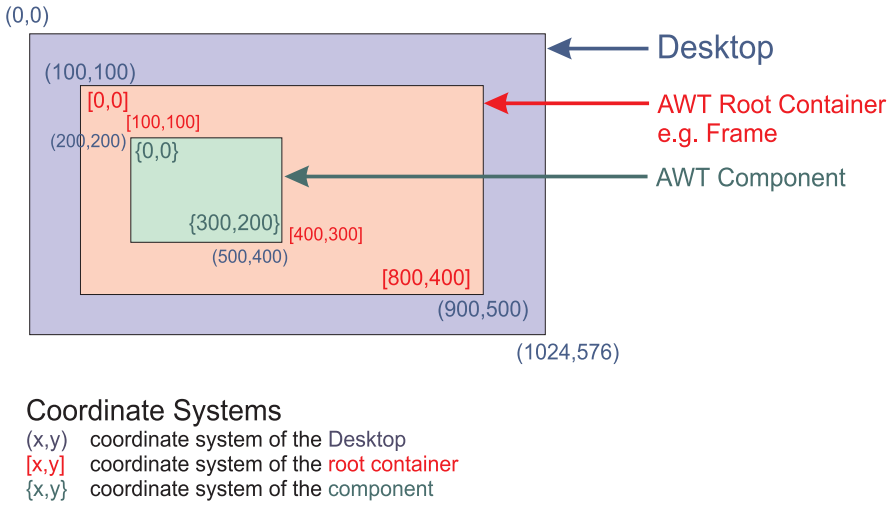


Figure 3 : AWT Coordinate system in a computer environment

Figure 3 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 3).

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.

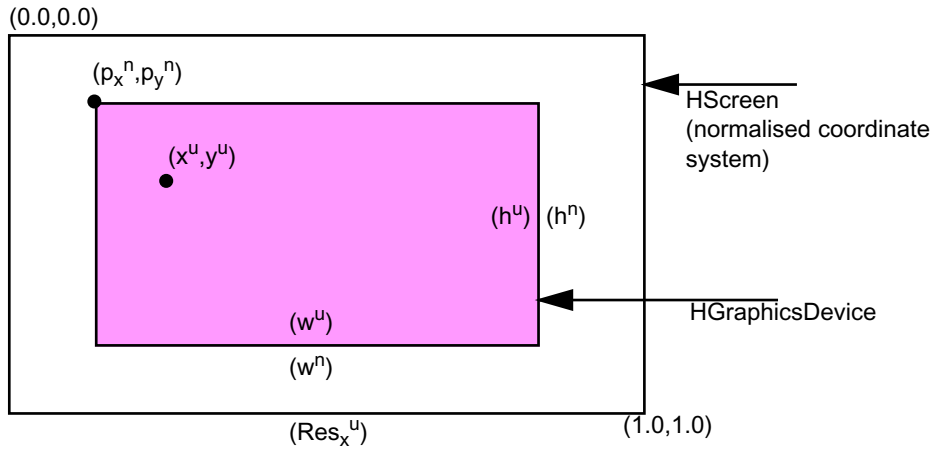


Figure 4 : Conversion between normalised and user coordinate systems

(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 x h^u than the point (x^u,y^u) in normalised coordinates is given by

$$x^n = \frac{x^{vu}}{Res_x{}^{vu}} \qquad y^n = \frac{y^{vu}}{Res_y{}^{vu}}$$

where

$$Res_x{}^{vu} = \frac{w^u}{w^n} \qquad Res_y{}^{vu} = \frac{h^u}{h^n}$$

is the virtual resolution of the `HScreen` in (sub)pixel and

$$x^{vu} = p_x{}^{vu} + x^u \qquad y^{vu} = p_y{}^{vu} + y^u$$

is the point (x,y) in the virtual coordinate space with

$$p_x{}^{vu} = p_x{}^n \cdot Res_x{}^{vu} \qquad p_x{}^{vu} = p_x{}^n \cdot Res_x{}^{vu}$$

being the location of the `HGraphicsDevice` in the virtual coordinate system.

Given the Point (x^n,y^n) the point (x^u,y^u) is given by

$$x^u = floor[(x^n - p_x{}^n) \cdot Res_x{}^{vu} + 0.5]$$

and

$$y^u = floor[(x^n - p_x{}^n) \cdot Res_x{}^{vu} + 0.5]$$

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

$$x^n = \frac{x^u}{w^u} \qquad y^n = \frac{y^u}{h^u}$$

and

$$x^u = x^n \cdot w^u \qquad y^n = y^n \cdot h^u$$

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.

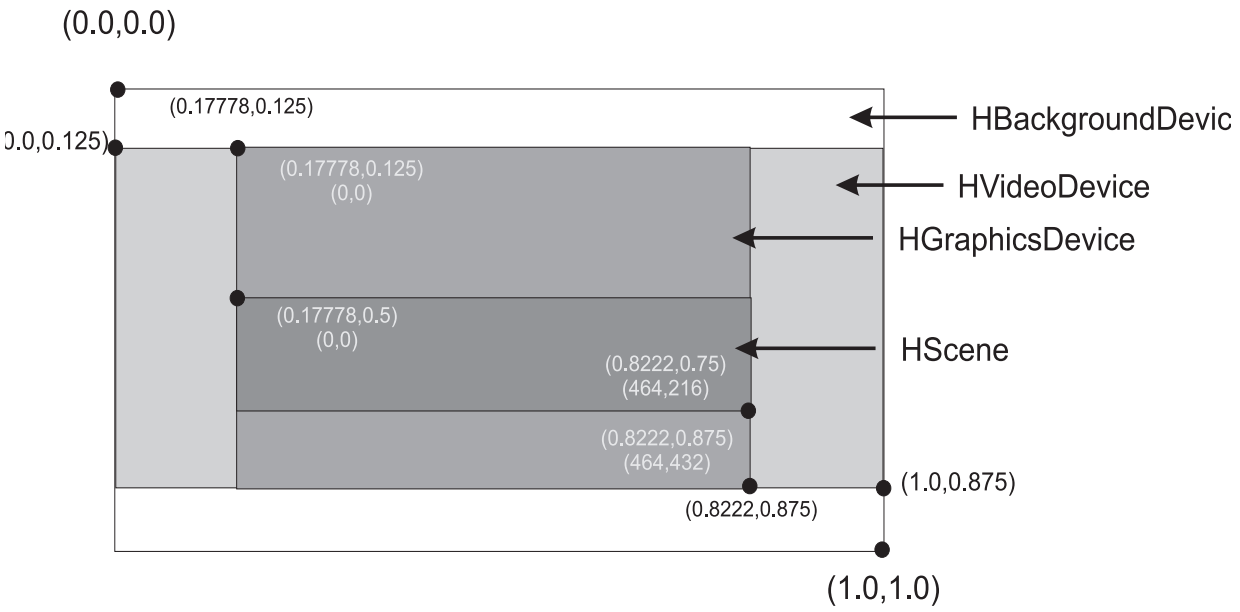


Figure 5 : Possible configuration of HAVi Devices

Figure 5 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 5 the `HGraphicsDevice` is configured to cover the area ($0.1\overline{7}$, 0.125) to ($0.8\overline{2}$, 0.875) with a pixel resolution of 464x432.

The `HScene` can be configured by setting the preference `HSceneTemplete.SCENE_PIXEL_RECTANGLE` with a Rectangle of (0,216,464,261) or by setting the preference `HSceneTemplete.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 6

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

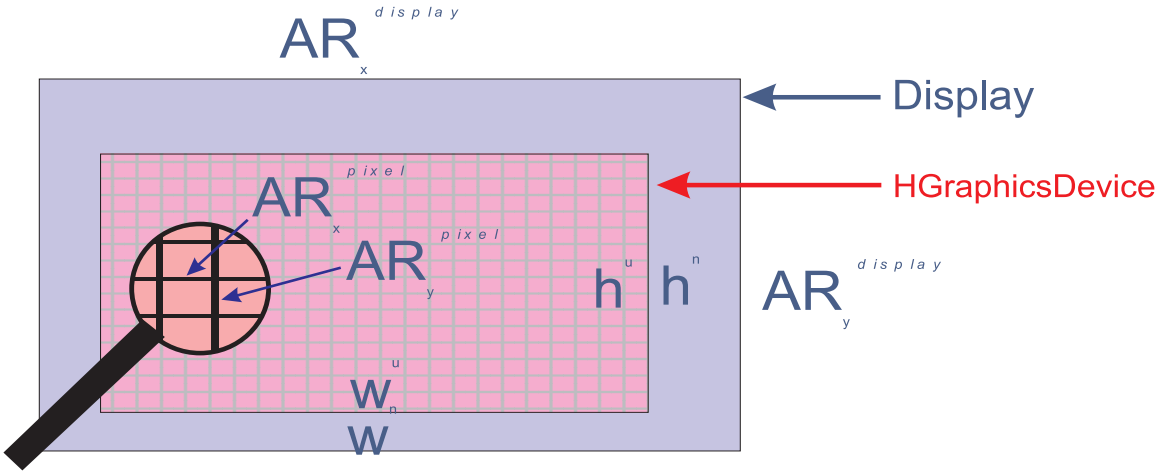


Figure 6 : Calculating the Pixel Aspect Ratio

Table 3 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 218.

Table 3 : Typical Resolutions and their pixel aspect ratio

	4:3 Display	16:9 Display
Resolution for full screen HGraphicsDevice	Pixel Aspect Ratio	Pixel Aspect Ratio
720x576	16:15	64:45
768x576	1:1	48:36
1024x576	36:48	1:1

13.2.1.4 Video space

The coordinate space for video is that defined by the input video signal after any scaling required by the platform (e.g. that required by ETR154). Scaling and clipping of video can be achieved using the JMF control `org.dvb.media.VideoPresentationControl`. The JMF control `VideoFormatControl` allows applications to query the various transformations being performed on video as part of its decoding and presentation.

13.3 Graphics

13.3.1 Modelling of the MHP display stack composition

The following sections describes the theoretical model of the MHP display stack. Unfortunately, certain real world constraints may apply see section 13.6, "Approximations" on page 158. The "Graphics, Video and Subtitle pipeline" is illustrated in figure 7.

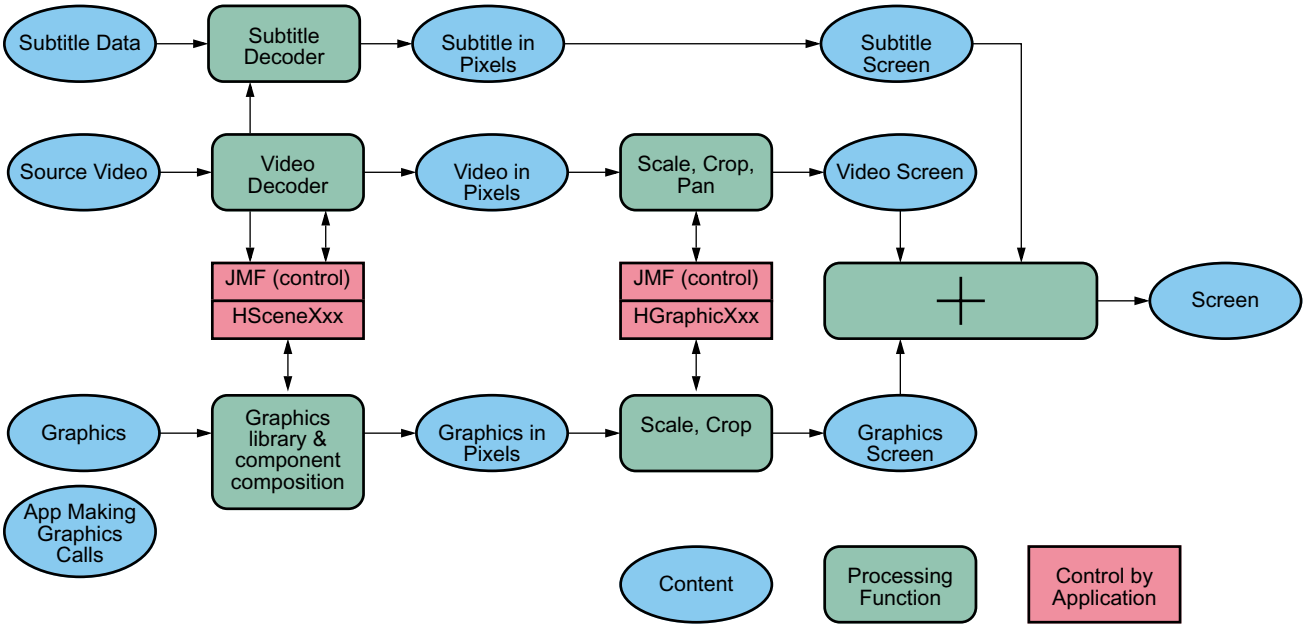


Figure 7 : Graphics, Video and Subtitle pipeline

From the application author's point of view the behaviour of the graphics composition process has 3 elements:

- The AWT elements are composed following the traditional graphics model using Porter-Duff rules (see Porter-Duff [D]). The default rule used is the SRC rule.
- The background and video planes are composed using the Porter-Duff rule SRC_OVER (Note that only alpha of 0 and 1 is used in the Video planes).
- The results are composed together using the SRC_OVER rule (with the AWT results as the source and the HAVi results as the destination).

This is illustrated in figure 8.

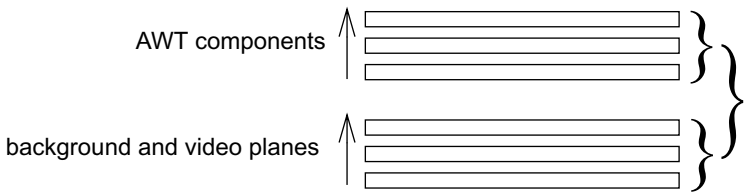


Figure 8 : Overview of AWT / HAVi plane composition

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 9) 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 HGraphicsVideoDevice will keep on playing when an application starts.

The stack is illustrated in figure 9.

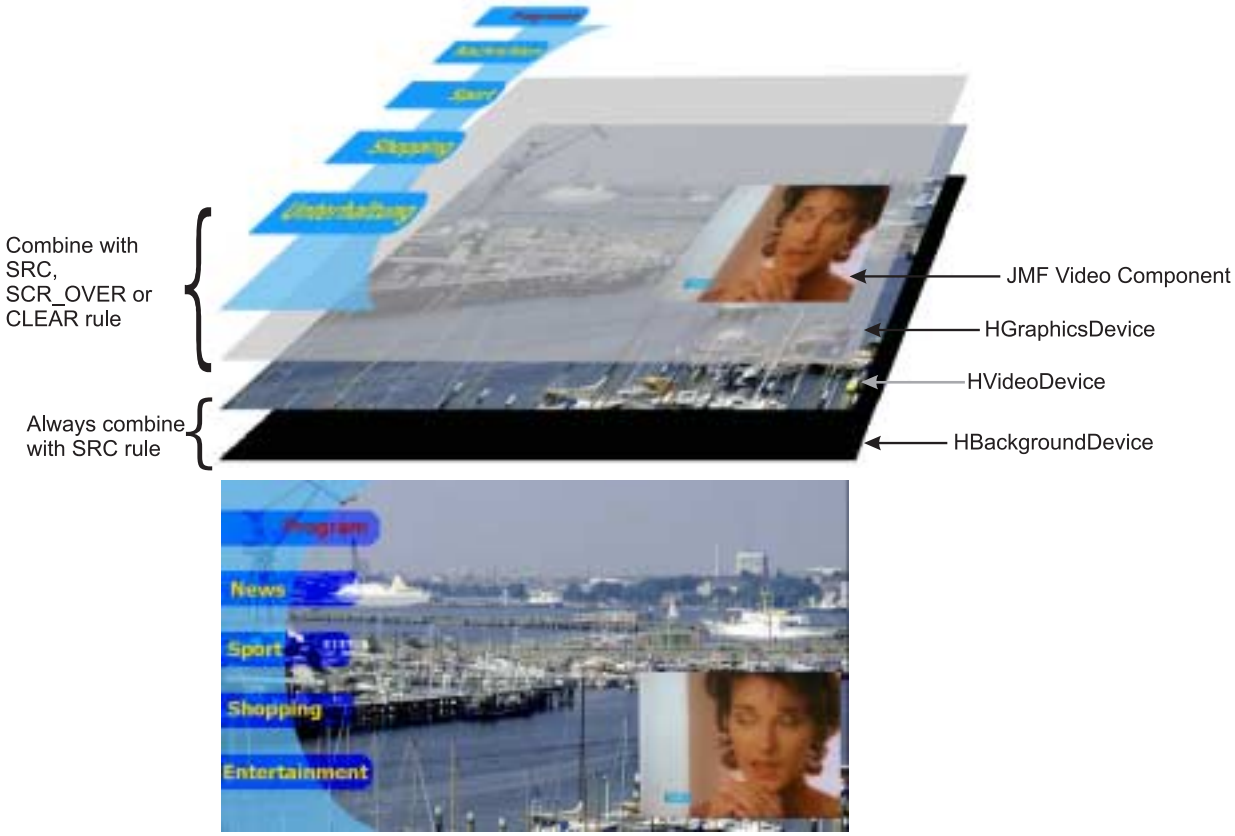


Figure 9 : The MHP display stack illustrated

13.3.2 AWT Reference Model in the MHP

In the AWT all graphics rendering of the lightweight components is done via the `java.awt.Graphics` class. When a component needs to be redrawn it issues a repaint command which gets passed from component to component from top to bottom until a heavyweight component is reached. Although there are no heavyweight components in the MHP the `HScene` can be thought of being similar to a heavyweight component. Thus in MHP devices the repaint command gets passed until it reaches the root container - the `HScene`. The repaint method of the `HScene` gets a `org.dvb.ui.DVB-`

`Grapahics` object cast to `java.awt.Graphics` from the underlying implementation and calls its paint method. The paint method of the `HScene` does no drawing, it only calls the paint methods of its children. Before calling the paint method of a child the origin gets translated to the coordinate system of the child and the graphics object passed is clipped to the size of the child (see figure10)

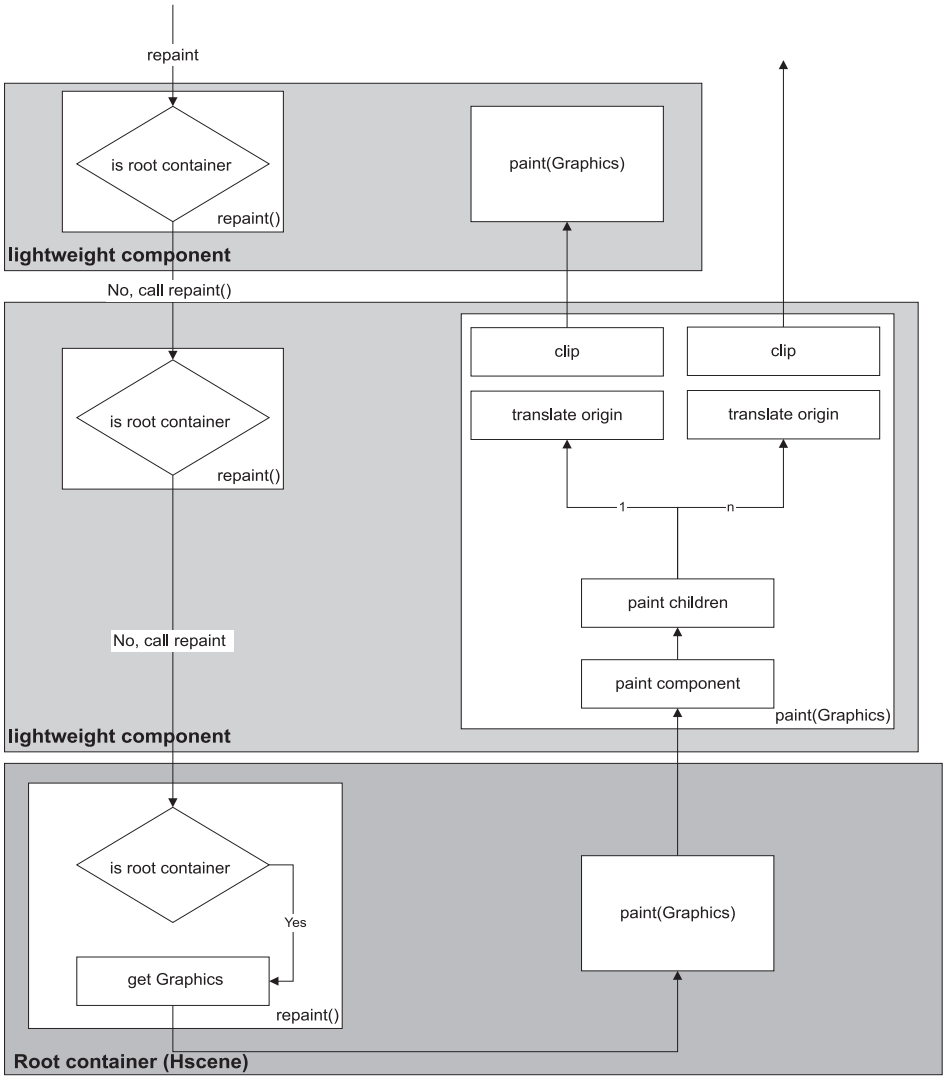


Figure 10 : Repaint model in the MHP

13.3.3 HAVi devices and AWT components

The top level user interface container for DVB-J applications is represented by the `org.havi.ui.HScene` class. DVB-J applications may obtain an instance of this class from an instance of `org.havi.ui.HSceneFactory`. The `HSceneFactory` class provides a number of methods which may be used to obtain an `HScene` depending on the specific requirements of the DVB-J application. `HScene` is conceptually equivalent to `java.awt.Window` or `java.awt.Frame` however it models the differing graphical environment found in many digital TV receivers. In this environment, there are typically significant constraints on what sizes and positions of top level containers may be possible.

One means for obtaining an `HScene` from an `HSceneFactory` is to populate an `HSceneTemplate` with a set of constraints and then request an `HScene` from the `getBestScene()` method which meets these constraints. Applications wishing to obtain a full screen sized `HScene` may use the `getFullScreenScene()` method specifying a particular graphics device on which the full screen scene should be presented. Applications wishing to ensure their graphics can cover a particular area of the screen may use the `getSelectedScene()` method to create an `HScene` meeting this requirement.

MHP terminals are allowed to support only non overlapping HScenes. In implementations not supporting overlapping HScenes following behaviour shall be implemented:

- Overlapping at creation or size change through HSceneFactory:

  returns a null reference if "REQUIRED" is used. Returns best effort if "PREFERRED" is used.

- Overlapping later when sizes change by awt:

  Size does not change. Fails silently.

The MHP specification provides a general model for video output devices using the model found in the HAVi specification. A final output video signal is expressed through the org.havi.ui.HScreen class and is the result of adding a number of different video components.

- · The output of one or more graphics decoders
- · The output of one or more video decoders
- · The output of any special decoders, e.g. a background

An abstraction of each of these decoders is represented by an instance of a class inheriting from HScreenDevice - HGraphicsDevice, HVideoDevice and HBackgroundDevice respectively. Each of these can potentially exist in a range of configurations which are exposed by instances of classes inheriting from HScreenConfiguration - HGraphicsConfiguration, HVideoConfiguration and HBackgroundConfiguration respectively. Where devices support multiple configurations, applications may construct and populate templates to define criteria to select between configurations. These templates are classes inheriting from HScreenConfigTemplate.

As well as the general features, some of these sub-classes provide support for specific features of the devices concerned. HGraphicsConfiguration supports loading of images and listing of fonts specific to that configuration. It also support conversion between various coordinate systems. HVideoDevice provides access to the source of the video currently being decoded (a Locator) and provides access to the decoder object for the video currently being decoded (a javax.media.Player). The HBackgroundClass provides a means to support MPEG I-frames through the HStillImageBackgroundConfiguration class.

The method HScreen.getCoherentScreenConfigurations (HScreenConfigTemplate[] configs) allows applications to express a common set of constraints for video, graphics and backgrounds and get back a coherent answer. In HGraphicsConfigTemplate, the constant VIDEO_MIXING allows applications to request configurations where graphics is super-imposed above video but without any requirement for pixels to be aligned. In HScreenConfigTemplate, there are constants to allow applications to ask for configurations as follows :-

- · VIDEO_GRAPHICS_PIXEL_ALIGNED - video & graphics pixels are the same size and aligned
- · ZERO_VIDEO_IMPACT - a graphics configuration must not change the existing video configuration
- · ZERO_GRAPHICS_IMPACT - a video configuration must not change the existing graphics configuration

13.3.4 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 11. 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 Java Class Libraries Vol. 1 [31]).

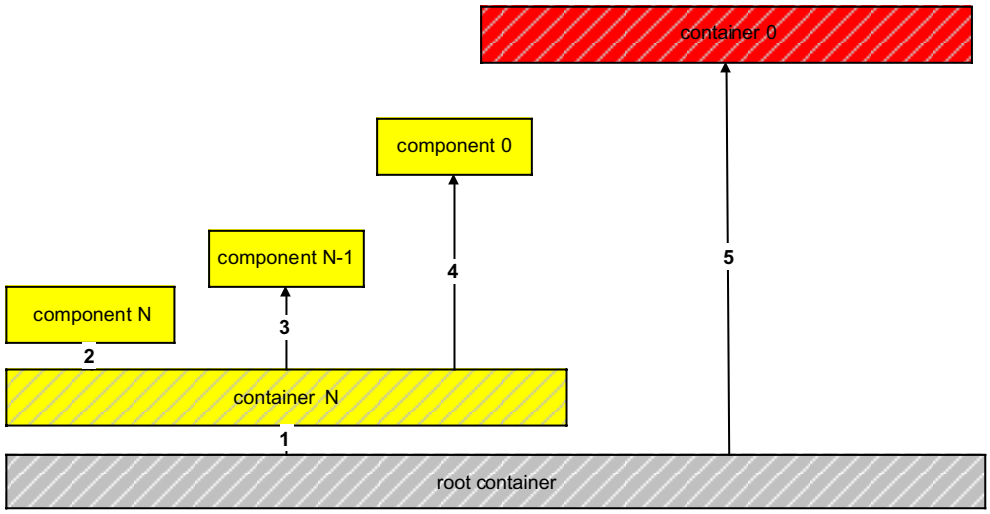


Figure 11 : Chronological order of painting

13.3.5 Composition Rules

13.3.5.1 Components generally

By setting an appropriate `DVBAlphaComposite` rule on a `DVBGraphics` and using translucent colours blending may be achieved. By repeated placement of components complex scenes can be described. In figure 12 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 12 : Summary of the Porter-Duff rules

13.3.6 Extensions to the AWT graphics capabilities

For graphics the MHP specification mainly uses AWT facilities defined in Java 1.1.x. However, certain extensions are made.

In order to allow compositing in the MHP the graphics capabilities of the JDK1.1 have been extended by providing the `org.dvb.ui` package.

The package consist of the classes `DVBGraphics`, `DVBAlphaComposite`, `DVBBufferedImage`, `DVBColor` and one Exception. See U, "(normative): Extended graphics APIs" on page 493.

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 JDK1.1 implementations each component gets passed a Graphics object in the rendering thread (e.g. in `public void paint(Graphics)`). In the MHP all graphics object will always be an instance of `org.dvb.ui. DVB-Graphics` cast to `java.awt.Graphics` (the same mechanism is used in Java 2 where all graphics objects are `java.awt.Graphics2D` objects cast to `java.awt.Graphics`). In implementations using Java 2 `DVBGraphics` will extend `java.awt.Graphics2D`.

`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 could support the SRC, CLEAR and SRC_OVER rule while an DVBGraphics Object created using a DVBBufferedImage of the type DVBBuffer-edImage.TYPE_BASE could only support the SRC and CLEAR rule. Application can query the available compositing rules using `DVBGraphics.getAvailableCompositingRules()`. When setting an unsupported compositing rule a `org.dvb.ui.UnsupportedDrawingOperationException` will be thrown.

The supported compositing rules are defined in G, "(normative): Minimum Platform Capabilities" on page 218.

The default compositing rule used by all graphics objects is SRC.

13.3.6.2 Buffered Image

The class `DVBBufferedImage` in the package org.dvb.ui adds the support for an accessible, transparent Image which can be used e.g. for off screen buffers. Two different platform dependent sample models are supported by `DVBBuffe-redImage`. When doing compositing in the `TYPE_BASE` sample model approximations may be applied (using SRC instead of SRC_OVER, see figure 20 or by approximating the alpha, see G, "(normative): Minimum Platform Capabili-ties" on page 218) 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.

The internal implementation of the AWT package shall use the `org.dvb.ui.DVBColor` class instead of the `java.awt.Color`. The class signatures shall not change. For example, where a method is specified to return `java.awt.Color` it shall return a `org.dvb.ui.DVBColor` cast to `java.awt.Color`.

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 13. 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.

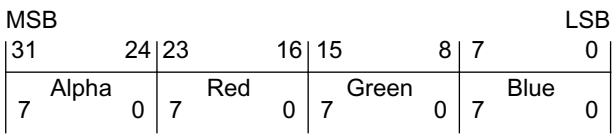


Figure 13 : MHP colour format (TYPE_INT_ARGB)

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.

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 14 illustrates the underlying format conversion control process in a ETR 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)

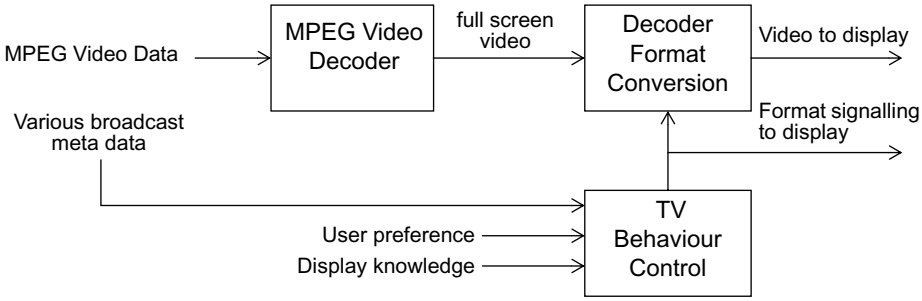


Figure 14 : Format control in TV mode

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

- Conventional TV format control behaviour (as in figure 14)
- 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.

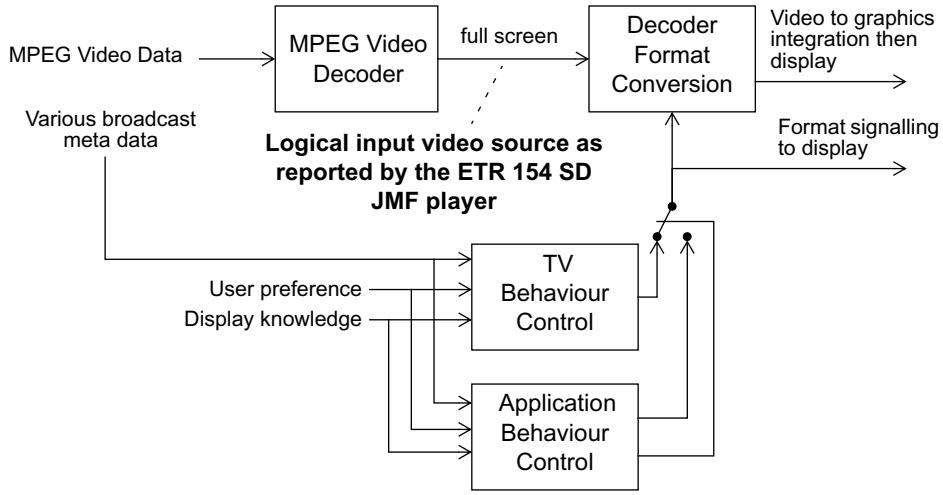


Figure 15 : Format control in the presence of a JMF player

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 16, these steps are clipping, scaling and positioning.

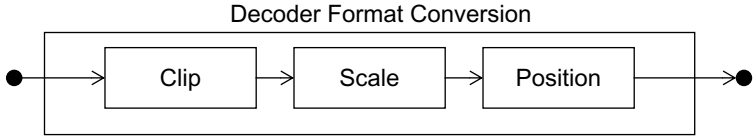


Figure 16 : Reference model for Decoder Format Conversion

An application can query the implemented capabilities of each step of the Decoder Format Conversion using the org.dvb.media.VideoPresentationControl ("VideoPresentationControl" on page 369). 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 374) that encapsulates the clipping, scaling, and positioning parameters. An application can get a number of pre-defined video transformations that correspond with standard Decoder Format Conversions like 16:9 letterboxing in a 4:3 display. It can either use these video transformations directly to set the Decoder Format Conversion, or it can change one or more parameters of the transformation before setting it. The API also offers support for querying the current video transformation.

13.4.3 Coordinate Spaces

The input to the Decoder Format Conversion block is always full screen video. If necessary, the MPEG video decoder block performs up-sampling as required by ETR154 to get full screen video.

An application expresses the video clipping in terms of the pixel-based coordinate space of the full screen video. For 50Hz SD video this is always a 720x576 raster.

Positioning of the video is expressed in the normalised coordinate space for background JMF players. For component-based JMF players, the position of the video component is expressed in the pixel-based device coordinate space (i.e., a video component is positioned like any other AWT component).

13.4.4 Video components

A scaled portion of a video can be presented as a component within the AWT hierarchy. The controls that influence this presentation are parts of AWT and JMF.

Video components are treated just like any other component. However, it shall be noted that pixels within the video have opaque colours. The mapping to the source video is provided by the video presentation control (see N, "(normative): Streamed Media API Extensions" on page 340) which allows an arbitrary portion of the video to be placed within the AWT hierarchy.

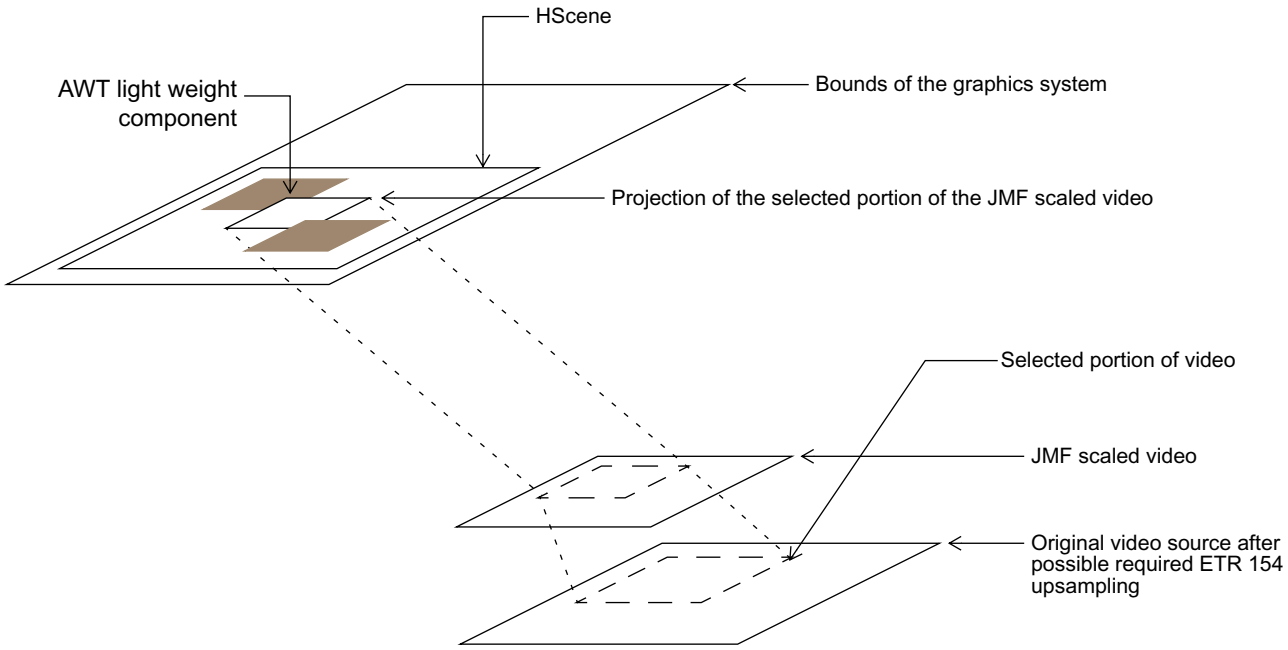


Figure 17 : Introducing video into the AWT component stack

13.5 Subtitles

13.5.1 Language and presentation setting

The following reference model shows the 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.

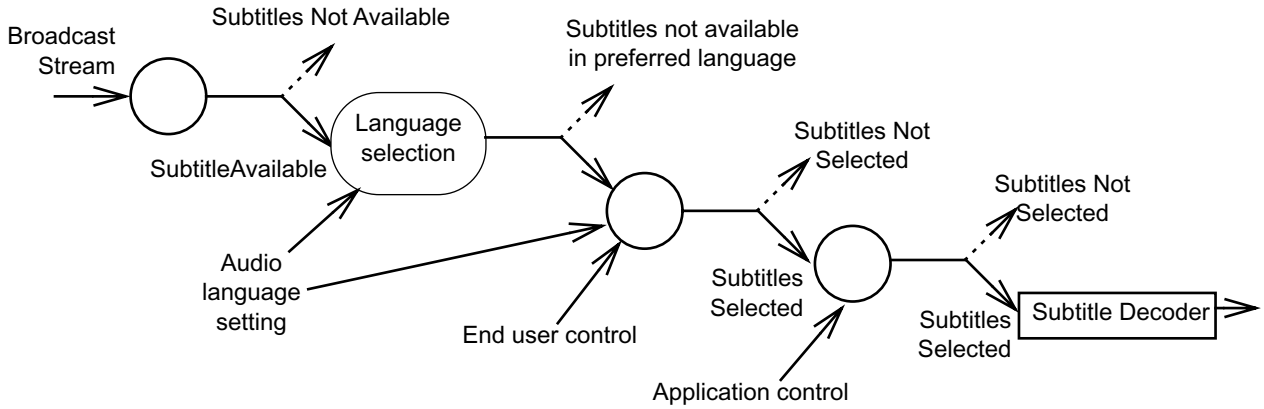


Figure 18 : Determining subtitling language and presentation setting

The DVB-J API includes a control (see "SubtitlingEventControl" on page 361) that the application can use to get notified of the state changes in the availability and presentation status of the subtitles. This corresponds to the status of the left hand side input and the right hand side output in the diagram.

The org.davic.media.SubtitlingLanguageControl allows the application to query the currently set subtitling language, override the default language setting and switch the subtitles off or let them be end user controlled. This corresponds to the language selection phase and the application controlled switch in the diagram

13.5.2 Relation to graphics

Ideally subtitles should be presented on top of the video plane but below the graphics plane(s). However, MHP terminals conforming to this specification are only required to support subtitles in areas where they are not overlapped by application graphics (i.e. by an `HScene` for a DVB-J application). The behaviour if the subtitles overlap with application graphics is non deterministic. Therefore, when presenting application graphics on screen, the application should either turn subtitles off or the broadcaster shall coordinate the use of screen area between subtitling and application graphics so that they will not overlap.

The subtitling plane is full screen and allows the subtitles to be positioned anywhere on screen. The positioning of the subtitling texts is part of the subtitling stream content and is fully controlled by the broadcaster.

13.5.3 Coordinate Spaces

Subtitles are decoded into a plane with the same coordinate system and position as the `HVideoDevice`.

13.6 Approximations

13.6.1 Approximations in composition

The MHP specification references the Porter-Duff rules for composition (see Porter-Duff [D]). The set of operations required is profile specific in addition the implementation of the compositions may be approximated in some profiles.

13.6.1.1 Implementation of modes

Typical implementations have a hardware blending process that blends the graphics plane over the video using SRC_OVER (see figure 19). This places certain practical limitations on the purity of the display model.

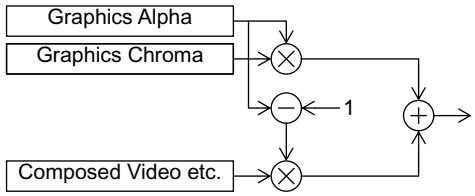


Figure 19 : Typical current technology implementation

The SRC rule simply requires that the colour and alpha characteristics of the new pixels replace those previously present.

`HVideoComponents` are treated as having an alpha of 1 so whether SRC or SRC_OVER is used when placing the video component the effect is that the video completely replaces anything previously drawn.

13.6.1.1.1 Graphics directly over video

When drawing graphics directly over a video component:

- The effect of SRC_OVER mode is as expected as the alpha value of the drawn graphics is used to control blending of the graphics with the video.

13.6.1.1.2 Graphics over other graphics

13.6.1.1.2.1 SRC

If (non-MPEG) graphics are painted over other (non-MPEG) graphics using SRC mode the result is as expected.

13.6.1.1.2.2 SRC_OVER

If (non-MPEG) graphics are painted over other (non-MPEG) graphics using SRC_OVER mode the result will be implementation dependant. Figure 20 on page 160 illustrates a variety of implementations of SRC_OVER graphics to graphics blending.

13.6.1.1.2.3 CLEAR

If (non-MPEG) graphics are painted over other (non-MPEG) graphics using the CLEAR mode the result is as expected. This operation is the same as using a source with alpha=0 and the SRC rule.

[a] Shows the logically correct result where the green and red areas mix to produce intermediate colours.

This implies a graphics to graphics blending process, it also implies a large gamut in both the chroma and alpha channels. This may not be practical in many early implementations.

The following cases illustrate simplifications of the blending scheme. These should not be considered equally good approximations:

[b] Here the graphics alpha is preserved only when it is drawn directly over a video component. When drawn over another (non-MPEG) graphic the alpha facets of the colours are considered to be 1.

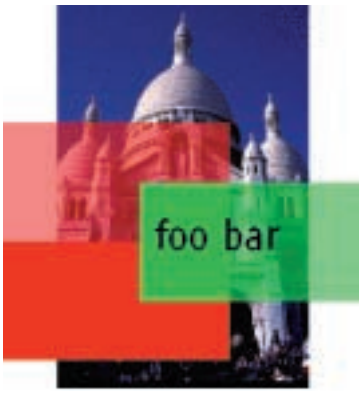
[c] Here the source graphics alpha is preserved when it is drawn over a video component even if there is an intermediate semi-transparent graphic. Where the source is over opaque graphics then a graphic to graphic blend is implemented.

[d] Here the source graphics alpha becomes the hardware mixing alpha regardless of what has been drawn previously over the MPEG image.

[e] Here the source graphics alpha becomes the hardware mixing alpha in areas where the alpha is already less than 1. Where the alpha is currently 1 (i.e. over opaque graphics) the alpha remains 1.

a) Perfect.

b) All graphics-graphics overlays are solid. Transparency is only preserved over video.



c) Transparency punches through transparent layers and blends with opaque layers.

d) Transparency is always to video and so punches through other graphics.

e) Transparency punches through transparent layers and graphics over solid graphics are not blended.

Figure 20 : Implementations of blending

13.6.1.2 Approximation of alpha

The precision of implementation of alpha depends on the `DVBGraphics` object concerned. The minimum requirements are specified in G, "(normative): Minimum Platform Capabilities" on page 218. 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 219.

14 System integration aspects

14.1 Namespace mapping

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 format shall be used:

dvb://<original_network_id>.[<transport_stream_id>][.<service_id>[.<component_tag>{&<component_tag>}][;<event_id>]]{/<path_segments>}

A more formal specification of the DVB URL expressed in BNF (as used in RFC 2396 [43]) is presented below:

```
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
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_segements as defined in RFC 2396 [43])
```

It should be noted that this syntax is fully compliant with the generic syntax of URIs as specified in RFC 2396 [43] and uses the registry-based naming authority version of that. Furthermore, all generic definitions specified in RFC 2396 [43] shall be valid for the DVB URL as well (e.g. escaping of special characters within file names, etc.).

RFC 2396 [43] 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 43).
- 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.

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.

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 41 + 15.1, "PNG - restrictions" on page 165	M	M	
	7.1.1.3, "PNG" on page 41 without restrictions	-	-	
	7.1.1.4, "GIF" on page 41	-	-	
	7.1.2, "MPEG-2 I-Frames" on page 41	M	M	
	7.1.1.2, "JPEG" on page 41	M	M	
Audio clips	7.1.4, "Monomedia format for audio clips" on page 43	M	M	
Video drips	7.1.3, "MPEG-2 Video "drips"" on page 41	M	M	
Text encoding	7.1.5, "Monomedia format for text" on page 43	M	M	
Broadcast streaming formats				
Video	7.2.2, "Video" on page 43	M	M	
Audio	7.2.1, "Audio" on page 43	M	M	
Subtitles	7.2.3, "Subtitles" on page 43	M	M	
Fonts				
Built in	Character set see annex E, "(normative): Character set" on page 212, Metrics see annex D, "(normative): Text presentation" on page 199 Face: UK RNIB 'Tiresias'	M	M	
Downloadable	7.4, "Downloadable Fonts" on page 44	M	M	
Broadcast channel protocols				
	6.2.2, "MPEG-2 Sections" on page 38	M	M	
	6.2.5, "DSM-CC User-to-User Object Carousel" on page 38	M	M	
	IP Multicast stack based on: 6.2.6, "DVB Multiprotocol Encapsulation" on page 39, 6.2.7, "Internet Protocol (IP)" on page 39 6.2.8, "User Datagram Protocol (UDP)" on page 39	O	Ro	M
Interaction channel protocols				
TCP/IP	6.3.3, "Transmission Control Protocol (TCP)" on page 40 6.3.2, "Internet Protocol (IP)" on page 39	-	M	M
UDP/IP	6.2.8, "User Datagram Protocol (UDP)" on page 39 6.3.2, "Internet Protocol (IP)" on page 39	-	M	M
DSM-CC U-U RPC	6.2.5, "DSM-CC User-to-User Object Carousel" on page 38 6.3.4, "UNO-RPC" on page 40 6.3.5, "UNO-CDR" on page 40	-	O	
HTTP	6.3.7, "Hypertext Transfer Protocol (HTTP)" on page 40	-	O	M
DVB-J				
Core	11.3, "Fundamental DVB-J APIs" on page 87	M	M	
Presentation	11.4.1, "Graphical User Interface API" on page 90	M (note 1)	M (note 1)	
	11.4.2, "Streamed Media API" on page 91	M	M	

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 94	M	M	
	11.5.2, "Support for Multicast IP over the Broadcast Channel" on page 95	O	Ro	
	11.5.3, "Support for IP over the Return Channel" on page 96	-	M	
	11.5.4, "MPEG-2 Section Filter API" on page 96	M	M	
	11.5.5, "Mid-Level Communications API" on page 96	-	M	
	11.5.6, "Persistent Storage API" on page 96	M	M	
Service Information & Selection	11.6.1, "DVB Service Information API" on page 96	M	M	
	11.6.2, "Service Selection API" on page 96	M	M	
	11.6.3, "Tuning API" on page 97	M	M	
	11.6.4, "Conditional Access API" on page 97	M	M	
	11.6.5, "Protocol Independent SI API" on page 97	M	M	
Common Infrastructure	11.7.1, "APIs to support DVB-J application lifecycle" on page 97	M	M	
	11.7.2, "Application discovery and launching APIs" on page 98	M	M	
	11.7.3, "Inter-Application Communication API" on page 98	M	M	
	11.7.4, "Basic MPEG Concepts" on page 99	M	M	
	11.7.5, "Resource Notification" on page 99	M	M	
	11.7.6, "Content Referencing" on page 99	M	M	
	11.7.7, "Common Error Reporting" on page 100	M	M	
Security	11.8.1, "Basic Security" on page 100	M	M	
	11.8.2, "APIs to Support TLS / SSL Over the Return Channel" on page 101	-	M	
	11.8.3, "Additional permissions classes" on page 101	M	M	
Others	11.9.1, "Timer Support" on page 101	M	M	
	11.9.2, "User Settings and Preferences API" on page 102	M	M	
	11.9.3, "Profile and version properties" on page 102	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

Engines are required to support ALL of the PNG colour types defines in PNG Specification Version 1.0 (see table 4). Engines are responsible for mapping these colours to those used by the engine's OSD.

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

Table 4 : PNG Formats

Colour Type	Allowed Bit Depths	Interpretation
0	1,2,4,8,16	Each pixel is a grayscale sample.
2	8,16	Each pixel is an R,G,B triple.
3	1,2,4,8	Each pixel is a palette index; PLTE chunk must appear.
4	8,16	Each pixel is a grayscale sample, followed by an alpha sample.
6	8,16	Each pixel is an R,G,B triple, followed by an alpha sample.

Where PNG graphics use colours defined in the currently active application palette these colours shall be reproduced correctly. Other colours shall be reproduced in an implementation dependent way.

Receivers should ignore gAMA (gamma) and cHRM (chromaticity) chunks in PNG files.

In this profile, all PNG graphics should be gamma corrected.

15.1.1 PNG Aspect ratios

PNG bitmaps shall carry a pHYs chunk indicating the pixel aspect ratio of the bitmap. This aspect ratio should be the same as that of the scene containing the bitmap.

The PNG specification indicates that if the aspect ratio is absent square pixels should be assumed. To avoid overriding this specification the aspect ratio should be signalled explicitly.

16 Registry of Constants

16.1 System constants

Table 5 : 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 101.
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 101.

Table 6 : Profile encoding

application profile	version			Definition
	major	minor	micro	
1	1	0	0	Enhanced Broadcast Profile 1 as defined in this specification
2	1	0	0	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.

Annex A (normative): Errata in external references

This section lists known errata in normative external references.

A.1 Java Class Libraries Vol. 1 [31]

A.1.1 java.lang.ThreadGroup.getParent()

This method may throw a security exception.

A.2 Java Class Libraries Vol. 2 [32]

A.2.1 java.awt.Component.getTreeLock()

The Java Class Libraries Second Edition, Volumes 1 and 2, ISBNs 0-201-31002-3

and 0-201-31003-1 contains an error in its description of this method.

It states that the locking object that is returned is for the entire AWT component tree. This is not required by this specification. The specification is as follows:

> Gets this component's locking object (the object that owns the thread synchronization monitor) for AWT component-tree and layout operations.
>
> Returns this component's locking object.

The specification's wording allows a different locking object to be returned for different component hierarchies, or even a different locking object for each component.

A.3 Java Language Spec. [33]

A.3.1 java.lang.ThreadGroup.getParent()

This method may throw a security exception.

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 [51] - 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.

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 188	This spec.
AdaptationLength	The MHP terminal may ignore the possible contents of the dsmccAdaptationHeader 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 privateData 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 ServiceGatewayInfo structure	DSM-CC
serverId	Shall be set to 20 bytes 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	32	uimsbf	+	
blockTimeOut	32	uimsbf	+	
minBlockTime	32	uimsbf	+	
taps_count	8	uimsbf	N1	≥ 1
{				
id	16	uimsbf	0x0000	user private
use	16	uimsbf	0x0017	BIOP_OBJECT_USE
assocTag	16	uimsbf	+	
selector_length	8	uimsbf	0x00	
}				
for (j=1; j<N1; j++) {				
id	16	uimsbf	+	
use	16	uimsbf	+	
assocTag	16	uimsbf	+	
selector_length	8	uimsbf	N2	Possible additional taps that may be ignored by MHP terminals.
for (j=0; j<N2; j++) {				
selector_data	8	uimsbf	+	
}				
}				
userInfoLength	8	uimsbf	N3	
for (k=0; k<N3; j++) {				
userInfo_data	8	uimsbf	+	
}				
}				

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 80).

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
label_descriptor() {				
descriptor_tag	8	uimsbf	0x70	
descriptor_length	8	uimsbf	N1	
for (n=0; n<N1; n++) {				
text_char	8	uimsbf		The label
}				
}				

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

text_char: The label that is attached to the module

B.2.2.4.2 Caching priority descriptor

To indicate priorities for the objects, a caching_priority_descriptor may be included in the userInfo field of the moduleInfo in the DownloadInfoIndication message.

This descriptor provides a priority value for the caching. The same priority applies for each object in the module. The priority indicated in the descriptor is only a hint to the MHP terminal and implementations may use that in combination with other caching strategies.

The descriptor includes also the transparency level (see section B.5.2, "Transparency levels of caching" on page 195) 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
caching_priority_descriptor() {				
descriptor_tag	8	uimsbf	0x71	
descriptor_length	8	uimsbf		
priority_value	8	uimsbf		
transparency_level	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 195.

Table B.10 : Transparency level values (Sheet 1 of 2)

Value	Description
0	reserved
1	Transparent caching
2	Semi-transparent caching

Table B.10 : Transparency level values (Sheet 2 of 2)

Value	Description
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()			+	See Table B.26 on page 184
downloadTaps_count	8	uimsbf	N1	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		
context_data_length	16	uimsbf	N3	
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.2.3 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.

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

Table B.13 : Restrictions on the BIOP Generic Object Message

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 174
objectKind_data id_data kind_data	32 8 8	Table B.19, “BIOP::DirectoryMessage syntax,” on page 176
objectKind_data	8	Table B.21, “BIOP::StreamMessage syntax,” on page 178
objectKind_data eventName_ data	32 8	Table B.23, “BIOP::StreamEventMessage syntax,” on page 180
type_id_byte	32	Table B.26, “IOP::IOR syntax,” on page 184
id_data kind_data	32 32	Table B.30, “Syntax of Lite Options Profile Body with ServiceLocation component.,” on page 187

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 - for DVB-J applications this descriptor can be ignored.	This spec.
MessageSubHeader:: ServiceContextList	The MHP terminal may skip the possible serviceContextList structures.	This spec.

Table B.16 : BIOP::FileMessage syntax

Syntax	bits	Type	Value	Comment
BIOP::FileMessage() {				
magic	4x8	uimsbf	0x42494F50	"BIOP"
biop_version.major	8	uimsbf	0x01	BIOP major version 1
biop_version.minor	8	uimsbf	0x00	BIOP minor version 0
byte_order	8	uimsbf	0x00	Big endian byte ordering
message_type	8	uimsbf	0x00	
message_size	32	uimsbf	*	
objectKey_length	8	uimsbf	N1	<= 4
for (i=0; i<N1; i++) {				
objectKey_data	8	uimsbf	+	
}				
objectKind_length	32	uimsbf	0x00000004	
objectKind_data	4x8	uimsbf	0x66696C00	"fil" type_id alias
objectInfo_length	16	uimsbf	N2	
DSM::File::ContentSize	64	uimsbf	+	objectInfo
for (i=0; i<N2-8; i++) {				
descriptor()	8	uimsbf	+	
}				
serviceContextList_count	8	uimsbf	N3	serviceContextList
for (i=0; i<N3; i++) {				
context_id	32	uimsbf		
context_data_length	16	uimsbf	N4	
for (j=0; j<N4; j++) {				
context_data_byte	8	uimsbf	+	
}				
}				
messageBody_length	32	uimsbf	*	
content_length	32	uimsbf	N5	
for (i=0; i<N5; i++) {				
content_byte	8	uimsbf	+	actual file content
}				
}				

B.2.3.4 Content type descriptor

Zero or one content type descriptors can be carried in the file `MessageSubHeader::ObjectInfo`. This optional descriptor identifies the media type of the file.

If this descriptor is absent or not sufficient to categorise the content type then the extension portion of the file name should be used be used to provide the media type mapping via table 4, "File type identification" on page 47.

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`	8	uimsbf	0x72	
`descriptor_length`	8	uimsbf		
`for (i=0; i<descriptor_length; i++) {`				
`content_type_data_byte`	8	uimsbf		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 [66] and hence `content_type_data` carries the payload of the Content-Type header defined in [66].

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 2)

Syntax	bits	Type	Value	Comment
BIOP::DirectoryMessage() {				
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	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	
for (i=0; i<N5; i++) {				Binding
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.26 on page 184
objectInfo_length	16	uimsbf	N9	
if(kind_data == 'fil'){				
DSM::File::ContentSize	64	uimsbf	+	0 means that file size is not signalled

Table B.19 : BIOP::DirectoryMessage syntax (Sheet 2 of 2)

Syntax	bits	Type	Value	Comment
for (j=0; j<N9-8; j++) {				
descriptor()			+	
}				
}				
else {				
for (j=0; j<N9; j++) {				
descriptor()			+	
}				
}				
}				
}				

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.2.4 Streams and 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.

B.2.4.1 BIOP StreamMessage

Table B.20 : Restrictions on the BIOP Stream Message

Field	Restrictions	Source
MessageSubHeader:: ObjectInfo	The ObjectInfo field contains the DSM::Stream::Info_T structure and optionally other data after the Stream Info structure. MHP terminals may ignore the DSM::Stream::Info_T structure and the possible other object info data following it. Broadcasts may set the duration field to zero to indicate undefined duration.	This spec.
MessageSubHeader:: ServiceContextList	The MHP terminal may skip the possible serviceContextList structures.	This spec.
MessageSubHeader:: MessageBody	The MessageBody carries a sequence of taps. There shall be at most one tap of use BIOP_PROGRAM_USE. This tap identifies the service that provides the media stream associated with the Stream object (via a deferred_association_tags_descriptor in the PMT). The tap may only reference programs that are broadcast on the same multiplex (i.e. MHP terminals shall not need to tune to a different multiplex in order to receive the referenced media stream). There shall also be at most one tap with use STR_NPT_USE, which MHP terminals shall interpret as described in ISO/IEC 13818-6 [26]. MHP terminals may ignore possible other Taps (such as BIOP_ES_USE).	This spec.

Table B.21 : 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 {		uimsbf		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.21 : BIOP::StreamMessage syntax (Sheet 2 of 2)

Syntax	bits	Type	Value	Comment
for (i=0; i<N4; i++) {				
context_id	32	uimsbf		
context_data_length	16	uimsbf	N5	
for (j=0; j<N5; j++) {				
context_data_byte	8	uimsbf	+	
}				
}				
messageBody_length	32	uimsbf	*	
taps_count	8	uimsbf	N6	
for (i=0; i<N6; i++) {				
id	16	uimsbf	0x0000	undefined
use	16	uimsbf	+	see Table 4-12 in DVB Guidelines for Data Broadcasting
assocTag	16	uimsbf	+	
selector_length	8	uimsbf	0x00	no selector
}				
}				

B.2.4.2 BIOP StreamEventMessage

Table B.22 : Restrictions on the BIOP Stream 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.20 on page 178 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: • 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.23 : 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	*	
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.23 : BIOP::StreamEventMessage syntax (Sheet 2 of 2)

Syntax	bits	Type	Value	Comment
data	8	uimsbf	+	see BIOP StreamMessage
}				
DSM::Event::EventList_T {				
eventNames_count	16	uimsbf	N4	
for (i=0; i<N4; i++) {				
eventName_length	8	uimsbf	N5	
for (j=0; j<N5; j++) {				
eventName_data	8	uimsbf	+	(including zero terminator)
}				
}				
}				
for (i=0; i<N2-(N3+10); i++) {				
objectInfo_byte	8	uimsbf	+	
}				
serviceContextList_count	8	uimsbf	N6	
for (i=0; i<N6; i++) {				
context_id	32	uimsbf		
context_data_length	16	uimsbf	N7	
for (j=0; j<N7; j++) {				
context_data_byte	8	uimsbf	+	
}				
}				
messageBody_length	32	uimsbf	*	
taps_count	8	uimsbf	N8	
for (i=0; i<N8; i++) {				
id	16	uimsbf	0x0000	undefined
use	16	uimsbf	+	see Table 4-12 in DVB Guidelines for Data Broadcasting
assocTag	16	uimsbf	+	
selector_length	8	uimsbf	0x00	no selector
}				
eventIds_count	8	uimsbf	N4	(= eventNames_count)
for (i=0; i<N4; i++) {				
eventId	16	uimsbf	+	
}				
}				

B.2.4.2.1 Stream event names and event ids

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.2.4.2.2 Association of event ids to event time

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.

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 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.2.4.3.3 Single firing of "do it now" events

MHP terminals shall respond at most one time to a "do it now" events delivered under a single table version.

B.2.4.3.4 Section number

For this specification MHP terminals shall only consider section number zero.

B.2.4.3.5 Stream event life time

The set of stream events described in the BIOP::StreamEventMessage is possibly a subset of the events that may be used by an application. Therefore MHP terminals shall accommodate the dynamic change BIOP::StreamEventMessage.

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.

B.2.4.3.6 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.24 : Encoding of table id extension for DSMCC_descriptor_lists

table_id_extension bits			Payload of DSM-CC section with table ID 0x3D
[15]	[14]	[13…0]	
0	0	eventID[13…0]	Section carries a single "do it now" event
0	1	xx xxxx xxxx	Section carries a single NPT reference descriptor
1	0	xx xxxx xxxx	Section carries one or more other stream descriptors. I.e - Stream event descriptor(s) with a future eventNPTs - Stream mode descriptor (can be ignored in this profile) - NPT endpoint descriptor (can be ignored in this specification)
1	1	reserved for future use	

B.2.4.3.6.1 Note:

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.2.4.3.7 Resources to monitor stream events

B.2.4.3.7.1 "do it now" events

"do it now" events are single shot events, accordingly MHP terminals need to make special efforts to ensure a high probability that they can be reliably received.

Broadcasters are responsible for placing all "do it now" stream descriptors that may be of interest to an application on a single PID. This may be the same PID as is used for other DSM-CC sections.

MHP terminals shall dedicate a section filter to monitoring the possible transmission of "do it now" events while there are any applications subscribed to these events.

B.2.4.3.7.2 scheduled events

The stream descriptors for scheduled events are transmitted several times in the period before the time that they should fire. This allows a high probability that they will be effective even if they are not monitored continuously by the MHP terminal.

Any scheduled stream event descriptors shall be transmitted at least once each second.

MHP terminals shall raise an event in response to a scheduled stream event provided that the stream event descriptors are broadcast for at least 5 seconds before the scheduled time.

B.2.4.3.7.3 number of NPT time bases

The MHP terminal is only required to monitor a single event time base (the NPT). So, even if an application invokes more than one `Stream` object as a source of `StreamEvents`, all of the implied DSM-CC StreamEventMessages shall specify the same tap as their source of NPTReferenceDescriptors (i.e. the taps with use STR_NPT_USE shall be the same).

B.2.4.3.8 Encoding of NPT time

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:

B.2.4.3.8.1 number range for NPT

The range of values used shall be in the range 0 to 0x0FFFFFFFF (which is unambiguous for both tcimsbf or uimsbf).

B.2.4.3.9 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 182).

Where the value of eventID identifies a "do it now" event then the value of eventNPT shall be ignored by the MHP terminal.

B.2.4.4 Stream Descriptors

B.2.4.4.1 NPT Reference descriptor

Except as follows MHP terminals shall interpret this descriptor as it is described in ISO/IEC 13818-6 [26]:

- MHP terminals may ignore the contentId field.

Summarising ISO/IEC 13818-6 [26] and commenting on its use in this specification:

- 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].

- NPT Reference descriptors shall be transmitted at least once per second.

B.2.4.4.2 NPT Endpoint descriptor

MHP terminals may ignore this descriptor if present.

B.2.4.4.3 Stream Mode descriptor

MHP terminals may ignore this descriptor if present.

B.2.4.4.4 Stream Event descriptor

The eventNPT field conveys the NPT value at which the event will occur (or has occurred).

MHP terminals shall ignore events where the eventNPT has passed.

The privateDataByte field does not need to be interpreted by the MHP terminal.

Note that an application can access the privateDataByte field via 11.4.2.3, "Extensions to the Framework" on page 92 and 11.5.1, "Broadcast Transport Protocol Access API" on page 94

See also "Encoding of NPT time" on page 183 and "Signalling of "do it now events"" on page 183.

B.2.4.5 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.25 : 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 shall ignore the object.	This spec.

Table B.26 : IOP::IOR syntax

Syntax	bits	Type	Value	Comment
IOP::IOR {				
type_id_length	32	uimsbf	N1	
for (i=0; i<N1; i++) {				
type_id_byte	8	uimsbf	+	Short alias type_id (e.g. "dir")
}				
taggedProfiles_count	32	uimsbf	N2	Profile bodies
IOP::taggedProfile()				For objects in broadcast carousels: either BIOPProfileBody or LiteOptionsProfileBody.
for (n=0; n<N2-1;n++) {				MHP terminal may ignore other profiles (2...N1) if present
IOP::taggedProfile()				
}				
}				

B.2.4.5.1 BIOPProfileBody

The BiopProfileBody is used for references to objects within the same object carousel.

Table B.27 : Restrictions on the BIOP Profile Body

Field	Restrictions	Source
BiopProfileBody::byte_ order	0 (indicating big-endian byte order)	DVB
BiopProfileBody:: LiteOptionComponents	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.28 : 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.28 : BIOP Profile Body syntax (Sheet 2 of 2)

Syntax	bits	Type	Value	Comment
use	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).
assocTag	16	uimsbf	+	
selector_length	8	uimsbf	0x0A	
selector_type	16	uimsbf	0x0001	
transactionId	32	uimsbf	*	
timeout	32	uimsbf	*	
}				
for (n=0; n<N4-18; n++) {				The MHP terminal may skip over the possible additional taps
additional_tap_byte	8	uimsbf		
}				
}				
for (n=0;n<N6;n++) {				N6=N1-2
BIOP::LiteComponent{				
componentId_tag	32	uimsbf	+	
component_data_length	8	uimsbf	N7	
for (i=0; i<N7; i++) {				
component_data_byte	8	uimsbf		
}				
}				
}				
}				

B.2.4.5.2 LiteOptionsProfileBody

The LiteOptionsProfileBody is used for making links to objects carried in other object carousels. The LiteOptionsProfileBody can be used to make references to objects carried in other carousels within the same transport or in other transport streams. In the API, when the LiteOptionsProfileBody is encountered, the application will get a ServiceXFRErrorEvent or a ServiceXFRException. The other carousel is never mounted automatically by the implementation, but the application may do so using the API.

Table B.29 : Restrictions on the Lite Options Profile Body

Field	Restrictions	Source
LiteOptionsProfileBody:: profile_data_byte_order	0 (indicating big-endian byte order)	DVB
LiteOptionsProfileBody:: LiteOptionComponents	The list shall contain a ServiceLocation component as the first component. The MHP terminal may ignore possible other components in the list.	This spec.
DSM::ServiceLocation	For objects carried in the broadcast object carousel, the service domain NSAP address shall follow the Carousel NSAP address format.	This spec.
DSM::ServiceLocation:: InitialContext	The MHP terminal may ignore the initial context	This spec.

Table B.30 : 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.31 "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.31 : DVB Carousel NSAP Address (Sheet 1 of 2)

Syntax	bits	Type	Value	Comment
DVBcarouselNSAPaddress()				
AFI	8	uimsbf	0x00	NSAP for private use
Type	8	uimsbf	0x00	Object carousel NSAP Address.

Table B.31 : DVB Carousel NSAP Address (Sheet 2 of 2)

Syntax	bits	Type	Value	Comment
carouselId	32	uimsbf	+	To resolve this reference a carousel_id_ descriptor with the same carousel_id as indicated in this field must be present in the PMT signalling for the service identified below.
specifierType	8	uimsbf	0x01	IEEE OUI
specifierData { IEEE OUI }	24	uimsbf	0x00015A	Constant for DVB OUI
dvb_service_location () {				
transport_stream_id	16	uimsbf	+	This may be set to 0x0000 which indicates that the MHP terminal shall not use the transport_stream_id when locating the service. For any other value then this field shall be used.
original_network_id	16	uimsbf	+	
service_id	16	uimsbf	+	(= MPEG-2 program_number)
reserved	32	bslbf	0xFFFFFFFF	
}				
}				

B.2.5 Assignment and use of transactionId values

The use of the transactionId in the object carousel is inherited from its use as defined by the DSM-CC specification, and as such it can appear somewhat complex. The transactionId has a dual role, providing both identification and versioning mechanisms for control messages, i.e. DownloadInfoIndication and DownloadServerInitiate messages. The transactionId should uniquely identify a download control message within a data carousel, however it should be "incremented" whenever any field of the message is modified.

Note: The term "incremented" is used in the DSM-CC specification. Within the scope of this specification this should be interpreted as "changed".

The object carousel is carried on top of one or more data carousels. By a data carousel used below the object carousel, we mean in this specification a set of DownloadInfoIndication message transmitted on a single PID and the DownloadDataBlock messages carrying the modules described in the DownloadInfoIndication messages. The DownloadDataBlock messages may be spread on other elementary streams than the DownloadInfoIndication messages. The DownloadServerInitiate message in the context of object carousels is considered to be part of the top level of the object carousel and not associated with any data carousel.

When a module is changed, the version number of the module needs to be changed. This implies that the DownloadInfoIndication message that references the module needs to be also updated. Since the DownloadInfoIndication is updated, the transactionId needs to be also changed. However, the transactionId of the DownloadInfoIndication message is used in other messages also, but the need to change the other messages should specifically be avoided and the implications of updating a module should be limited to the module itself and the DownloadInfoIndication that references the module. Therefore, additional rules on the usage of the transactionId have been specified as follows.

The transactionId has been split up into a number of sub-fields defined in Table B.32. This reflects the dual role of the transactionId (outlined above) and constraints imposed to reduce the effects of updating a module. However, to increase interoperability the assignment of the transactionId has been designed to be independent of the expected filtering in target MHP terminals.

Table B.32 : Sub-fields of the transactionId

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 must be incremented/changed every time the control message is updated.
30-31	Bit 30 - zero Bit 31 - non-zero	Originator	This is defined in the DSM-CC specification [26] as 0x02 if the transactionId has been assigned by the network - in a broadcast scenario this is implicit.

Due to the role of the transactionId as a versioning mechanism, any change to a control message will cause the transactionId of that control message to be incremented. Any change to a Module will necessitate incrementing its moduleVersion field. This change must be reflected in the corresponding field in the description of the Module in the DownloadInfoIndication message(s) that describes it. Since a field in the DownloadInfoIndication message is changed its transactionId must be incremented to indicate a new version of the message. Also, any change in the DownloadServerInitiate message implies that its transactionId must also be incremented. However, when the transactionId is divided into subfields as specified above, updating a message will change only the Version part of the transactionId while the Identification part remains the same.

Since the transactionId is used also for identifying the messages when referencing the messages in other structures, it is very desirable that these referenced would not need to be updated every time the control message is update. Therefore the following rule shall be applied when locating the messages based on the references:

When locating a message based on the transactionId value used for referencing the message, only the Identification part (bits 1...15) shall be matched.

Using this rule, the implications of updating a module can be limited to the module itself and the DownloadInfoIndication message describing the module. Also, this implies that if an MHP terminal wants to find out if a particular module that it has retrieved earlier has changed, it needs to filter the DownloadInfoIndication message that described that module and check if it has been changed.

B.2.6 Mapping of objects to data carousel modules

The DSM-CC Object Carousels allow one or more objects to be carried in one module of the data carousel. In order to optimize the performance and memory requirements two additional requirements are specified:

- When mapping objects to modules of a data carousel, only closely related objects should be put into one module. Objects that are not closely related should not be put into the same module. If in the process of retrieving an object from the carousel an MHP terminal acquires a module containing multiple objects, it should attempt to cache these since the expectation should be that the other objects are related to the object requested and probably will be needed soon.

- The size of a module that contains multiple objects should not exceed 65536 bytes when decompressed[1]. MHP terminals complying to this specification are only required to handle modules containing multiple objects where the module size when decompressed is 65536 bytes or less. Modules containing a single file message can exceed 65536 bytes with upper size only limited by the memory resources in the MHP terminal.

- In addition to the limitations imposed by the 65536 byte limit, directory and service gateway messages are limited to 512 object bindings per message.

B.2.7 Compression of modules

The modules may be transmitted either in uncompressed or compressed form. If the module is transmitted in compressed form, this is signalled by including the compressed_module_descriptor in the userInfo field of the moduleInfo in the DownloadInfoIndication message.

Presence of the compressed_module_descriptor indicates that the data in the module has the "zlib" structure as defined in RFC1950.

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	
}			

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

1. I.e. when the file has been decompressed from the file transport but before the content decoding has started.

B.2.8 Mounting an Object Carousel

The ServiceGateway object is the root directory of the file system delivered by an Object Carousel and must be acquired before any other object can be downloaded. This may be achieved by two compatible mechanisms. The signalling of which mechanisms are being supported by a broadcast is provided by the carousel_id_descriptor. This descriptor may be included in the second descriptor loop of a PMT corresponding to a PID on which the DSI message for an Object Carousel is broadcast, i.e. the boot-PID.

In this specification the use of the carousel_id_descriptor for signalling is mandatory. The consequence is that if a PMT second descriptor loop contains a data_broadcast_id_descriptor that provides signalling for this specification, it shall also contain a carousel_id_descriptor.

Note: A single PID shall only contain messages from a single Object Carousel and so only one carousel_id_descriptor shall be present in any second descriptor loop. However, a single service may contain more than one Object Carousel. Consequently, the carousel_id_descriptor may appear more than once in any single PMT.

The acquisition of the ServiceGateway object may be via the standard DSI-DII mechanism. This shall be supported by all broadcasts regardless of signalling in the carousel_id_descriptor and shall be sufficient for all MHP terminals.

See also 10.2, "Program Specific Information" on page 64.

A broadcast may also contain additional information in the carousel_id_descriptor to support the "enhanced" boot mechanism. This is signalled by setting the formatId field for this descriptor to 0x01. This additional information is an aggregation of all the fields necessary to locate the ServiceGateway, also found in the DSI and DII messages. However, in such a case the module containing the ServiceGateway object shall be broadcast on the PID identified by the data_broadcast_id_descriptor. It is optional for both broadcasts and MHP terminals to support this mechanism.

B.2.8.1 carousel_id_descriptor

This descriptor is MPEG defined and in this specification may be included in the second descriptor loop of a PMT.

Table B.35 : Carousel identifier descriptor syntax

Syntax	bits	Type	Value
carousel_identifier_descriptor {			
descriptor_tag	8	uimsbf	0x13
descriptor_length	8	uimsbf	N1
carousel_id	32	uimsbf	
FormatID	8	uimsbf	
if(FormatID == 0x00) {			
for(i=0; i<N1-5; i++){			
private_data_byte	8		
}			
}			
if(FormatID == 0x01) {			
ModuleVersion	8	uimsbf	
ModuleId	16	uimsbf	
BlockSize	16	uimsbf	
ModuleSize	32	uimsbf	
CompressionMethod	8	uimsbf	
OriginalSize	32	uimsbf	
TimeOut	8	uimsbf	
ObjectKeyData	8	uimsbf	N2 ≤ 4
for(i=0; i<N2; i++){			
ObjectKeyData	8	bslbf	
}			
for(i=0; i<N1-N2-21; i++){			
private_data_byte	8		
}			

Table B.35 : Carousel identifier descriptor syntax

Syntax	bits	Type	Value
}			
}			

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 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.3 AssociationTag Mapping

B.3.1 Decision algorithm for association tag mapping

The following figure illustrates the decision tree for identifying the elementary stream(s) by which the object carousel is distributed:

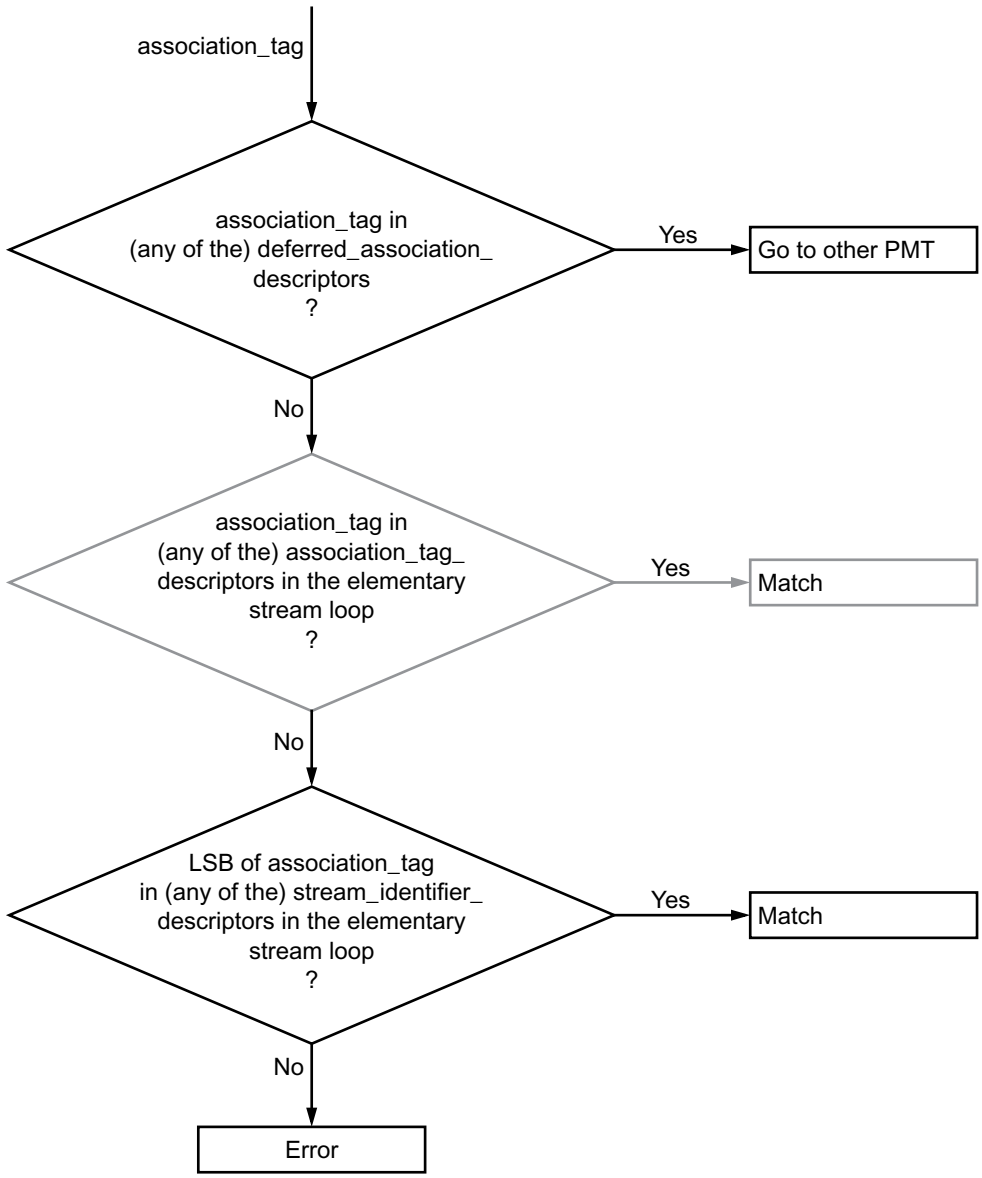


Figure B.1 : Object Carousel ES identification decision tree

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.3.2 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, assocation_tags are used by DSM-CC in order to refer to an elementary stream without directly referencing a PID value. An assocation_tag value is mapped to an elementary stream by matching the LSB of the assocation_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 assocation_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 assocation_tag value in an assocation_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 assocation_tag_descriptors.

The deferrered_assocation_tag_descriptor, as defined by ISO/IEC 13818-6 [26] section 11.4.3, is used to refer an association_tag to a different PMT (i.e. a different service). When attempting to map an assocation_tag to an elementary stream the assocation_tag must first be checked against any deferred_association_tag_descriptors in the current PMT (current in this context means the PMT of the service within which the assocation_tag is being mapped). If the association_tag matches any of the association_tags present in a deferred_assocation_tag_descriptor then the matching process restarts in the service indicated in that descriptor. If the transport_stream_id field in the deferred_association_tag_descriptor is set to 0x0000 then it shall be ignored and the MHP terminal is free to choose which transport stream ID it selects when obtaining a service.

B.3.3 deferred_association_tag_descriptor

The transport_stream_id field may take value 0x0000 in which case it shall be ignored in resolving this reference.

B.4 Example of an Object Carousel (informative)

The figure below illustrates an object carousel that is distributed over three elementary streams belonging to the same service.

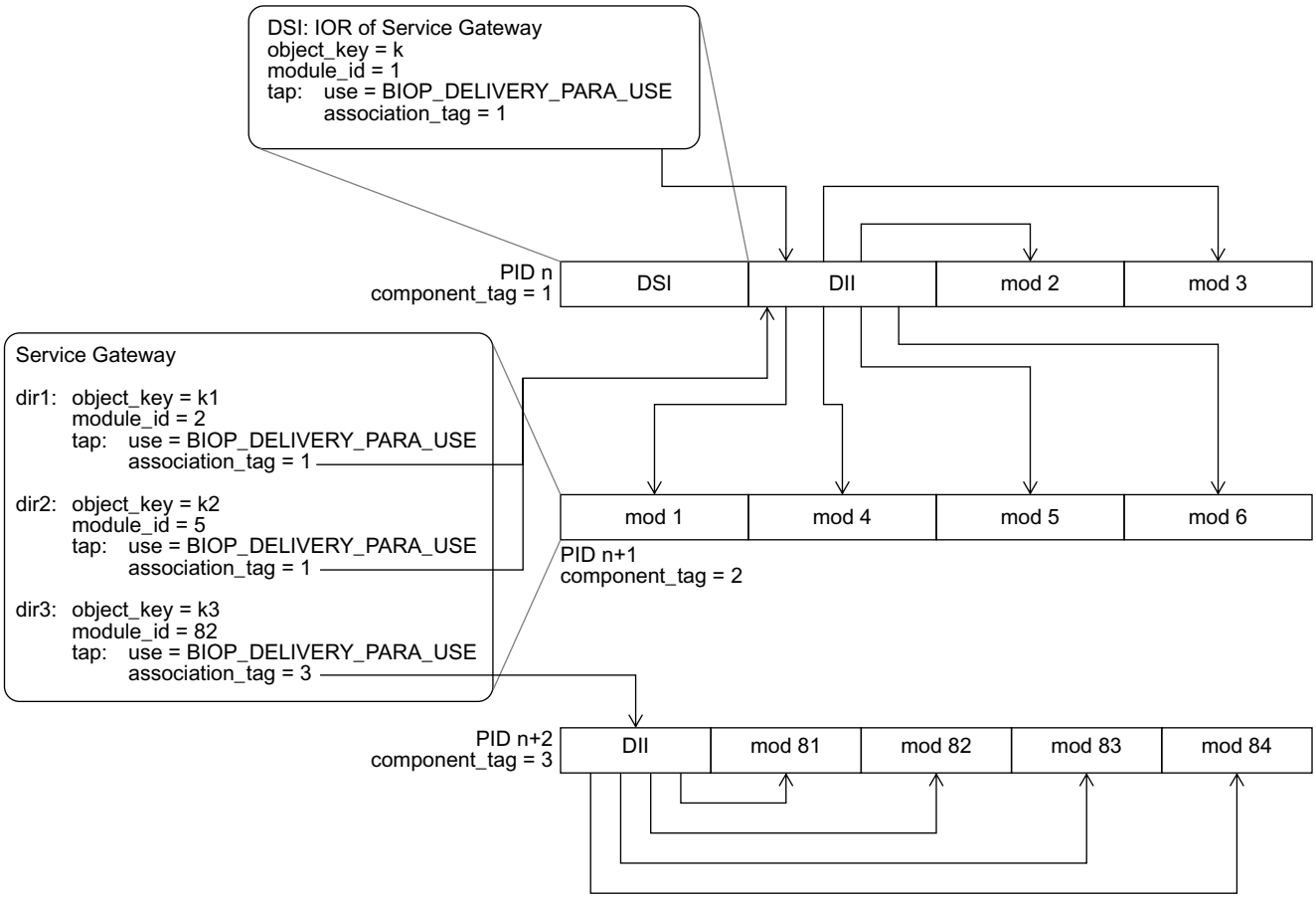


Figure B.2 : Example carousel

The DownloadServerInitiate (DSI) message is carried on the first elementary stream. It contains the object reference that points to the ServiceGateway. The tap with the BIOP_DELIVERY_PARA_USE points to a DownloadInfoIndication (DII) message that provides the information about the module and the location where the module is being broadcasted. In the example, the ServiceGateway object is in the module number 1 that is carried on the second elementary stream (indicated by a BIOP_OBJECT_USE tap structure in the DII message).

The ServiceGateway object is a root directory that, in this example, references three subdirectories. Taps with BIOP_DELIVERY_PARA_USE are used in the object references of the subdirectories to provide links to the modules via the DownloadInfoIndication (DII) message. The two first subdirectories "dir1" and "dir2" are referenced in the DII message that is carried in the first elementary stream. The third subdirectory is referenced in the DII message carried in the third elementary stream.

In this example, the two first elementary streams carry the messages of one logical data carousel while the third elementary stream carries the messages of another logical data carousel. All these belong to the same object carousel. In the example, the third elementary stream contains the objects in the "dir3" subdirectory and the objects in the "dir1" and "dir2" subdirectories are distributed over the first and second elementary stream.

It is important to note that the third elementary stream may originate from a completely separate source than the first two elementary streams. The directory hierarchy and objects contained in the third elementary stream are "mounted" in the root directory by providing the "dir3" directory entry with the appropriate location information.

This type of structure could be used, for example, in a national information service that contains some regional parts. The common national parts could be carried in this example case on the two first elementary streams that are distributed unmodified in the whole country. The regional parts are carried in the third elementary stream that is locally inserted at each region. From the application's point of view, the common national parts are in the "dir1" and "dir2" subdirectories while the regional parts are in the "dir3" subdirectory.

Another example where this type of structure could be used is if the service contains multiple independent applications. In this case, each application could be placed in its own subdirectory and these subdirectories might be carried as separate data carousels on different elementary streams.

B.5 Caching

This section describes the constraints that an MHP terminal compliant with this specification shall implement when caching any content from the object carousel in the memory of the MHP terminal. Caching is optional for the MHP terminal, but if implemented shall conform to the constraints set in this section.

B.5.1 Determining file version

There is no version number directly related to files (or other BIOP messages), the closest association is the moduleVersion in the DII that references the module that contains the BIOP message.Therefore,to ensure that a file is up to date the MHP terminal must determine that the moduleVersion for the appropriate module is current and reacquire if necessary.

When this checking is required is defined by the transparency level as specified in the following section.

B.5.2 Transparency levels of caching

The definition of transparency levels describes the behaviour that the MHP terminal shall implement when the content in the object carousel is changing. The transparency level determines how certain the MHP terminal is required to be about the validity of the content when returning the content to the application. The object carousel provides a mechanism for determining version changes of the content by monitoring the DII messages.

Validity of content is specified here in terms of the version number of the module that is broadcast in the DII message. The contents of an object as cached in the memory of the MHP terminal are defined to be valid at a certain point in time when the version number of the module in the cache matches the version number of the module as signalled in the DII message describing that module as it was last broadcast. Note that the definition is based on the DII message that was last broadcast and it may be that the MHP terminal was not filtering for this message at that time and did not receive it.

From the MHP terminal point of view, the transparency level indicates the constraints that the terminal needs to implement for monitoring the DII messages.

The broadcaster can indicate the appropriate transparency level that shall be applied for a given piece of content by using a descriptor associated with a module in the DII message (see "Caching priority descriptor" on page 171). 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 organize its caching strategy. One possible strategy is so-called active caching. This means that the terminal has a dedicated section filter for eachDII 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 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.	

Annex D (normative): Text presentation

D.1 Scope

This section addresses the following topics:

- How downloaded fonts are associated with applications and accessed by them
- The DVB-J APIs that are used for presenting text and their behaviour

Two levels of interface are addressed:

- Simple string rendering as supported by `java.awt.Graphics.drawString`
- More complex text object rendering as supported by DVB Text Layout Manager as described in U, "(normative): Extended graphics APIs" on page 493.

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 212.
- For the font families, sizes, styles and weights supported by implementations see and the presentation of this to the API G.4, "Resident fonts" on page 220.
- For the content formats used to deliver fonts see 7.4, "Downloadable Fonts" on page 44.

D.2 Fonts

D.2.1 Embedded fonts

See G.4, "Resident fonts" on page 220.

D.2.2 Downloaded fonts

D.2.2.1 Font technology

See 7.4, "Downloadable Fonts" on page 44.

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

```
<!ENTITY % fontdirectory SYSTEM "fontdirectory.en">
   <!-- the main entity for the font direcotry -->

<!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"
>
```

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 directory 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 MHP
   fontdirectory//EN"
   "http://www.dvb.org/DTD/MHP/fontdirectory.xml">
   <fontdirectory>
      <font>
         <name>Tiresias</name>
         <fonttype>PFR</fonttype>
         <filename>tiresias.pfr</filename>
         <style>BOLD</style>
      </font>
      <font>
         <name>Broadcaster X Screen Font</name>
         <fonttype>PFR</fonttype>
         <filename>brxsf.pfr</filename>
      </font>
   </fontdirectory>
```

D.2.2.3 Name and location of font index files

D.2.2.3.1 General

The specification of font paths and fonts by an application are private to that application, they are not available to other applications.

If the application creates a Font object with a *font face name* that is the same as one of the receiver's resident fonts then the specified font is used instead of the resident font with the same name. So, a downloaded font with *font face name* of "Tiresias" or "SansSerif" overrides any resident font with that name (see G.4, "Resident fonts" on page 220).

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 the base directory of the application.

The referenced font files can be on other file systems. However, the other file system must have already been mounted for the file loading to be successful. Font loading problems caused by referencing an unmounted file system are handled with the behaviour specified in java.AWT for an unavailable font.

D.2.2.4 Specification of fonts at run time

D.2.2.4.1 DVB-J

The implementation locates the font file using the parameters passed to the `java.awt.Font` constructor.

```
public Font(String name,
                      int style,
                      int size)
```

The font file is located by searching the directory for a file where the name element matches the name parameter of the `Font` constructor, a `style` element matches the style parameter and the `size` parameter is within the limitations in the size element.

It is font format specific how the font information for a given style and size is encoded in the font file.

D.3 Text rendering

D.3.1 Philosophy

This section describes "logical" rules that ensure text flows predictably on all receivers and defines some rendering requirements to ensure that a minimum acceptable level of text legibility is achieved.

No restriction is placed on the rendering technology used in a receiver provided that it achieves the deterministic text flow characteristics and the minimum rendering requirements described in this section.

D.3.2 Low and high level rendering

Two levels of interface are addressed:

D.3.2.1 Low level rendering

Simple string rendering as supported by:

- `java.awt.Graphics.drawString`
- `java.awt.Graphics.drawChars`
- `java.awt.Graphics.drawBytes`

This is referred to as "low level" rendering implying that the application author has significant responsibilities for ensuring that the text is visible. This rendering obeys the normal AWT rules. For example, the author is responsible for placing individual words or lines of text on to a component.

D.3.2.2 High level rendering

More complex text object rendering as supported by:

- `org.dvb.DVBTextLayoutManager`

This is referred to as "high level" rendering implying that the application author may need less effort to ensure that the text is visible. For example, the author could use the text layout manager to handle flowing paragraphs of text into an `org.havi.ui.HText` object

D.3.3 Font Definition

The nomenclature used in this section is derived from the resident font format(s). The nomenclature and the numerics provided here are directly applicable to downloaded fonts.

Font bounds

The font definition induces a set of parameters `xMin`, `xMax`, `yMin` and `yMax` that are **properties of the font**. These define the maximum extent of the outline representation of characters within the physical font, and as such are defined in terms of outline resolution units (outlineResolution).

(xMin, yMin) and (xMax, yMax) are the bottom-left and top-right corners of an imaginary bounding rectangle within which all characters in the font can be completely enclosed.

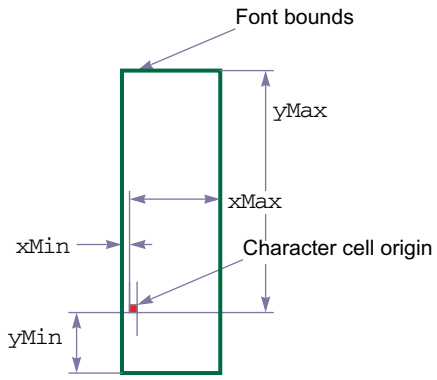


Figure D.1 : Font bounds

In "Low level rendering" the author is responsible for using knowledge of these values to correctly position text. The following Java methods provide access to these parameters:

- `java.awt.FontMetrics.getMaxAscent` derives from `yMax`
- `java.awt.FontMetrics.getMaxDescent` derives from `yMin`
- `java.awt.FontMetrics.getMaxAdvance` derives from `xMax`

In each case the value returned is that in the font definition converted into the AWT device pixels. Typically there is a loss of precision in this process as the font is typically defined at higher resolution than the display device. In addition, precision may be lost if the aspect ratio of the font coordinate system is different from that of pixels in the display.

`java.awt.FontMetrics.stringWidth` contains the summated string widths of all of the characters in the string taking into account `xMin` & `xMax` plus any adjustments in the font such as kerning.

All approximations shall round up (see D.3.5, "Rendering within limits" on page 204).

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" on page 204.

D.3.3.1 "Physical" font data

"Physical" font data such as horizontal escapement and kerning is defined in terms of metrics resolution units (metricsResolution). This is a high resolution representation, abstracted from any actual rendering system.

NOTE: The outlineResolution and metricsResolution are not necessarily the same.

D.3.4 Converting font metrics to display pixels

Many of the calculations in this section are in a high resolution physical coordinate system, either metrics or outline resolutions. These values need to be converted into the pixel resolution of the HGraphicsDevice to allow characters to be rendered.

Values in terms of these high level resolutions can be simply converted to values in terms of points by multiplying by the font size (in points) and dividing by the resolution, i.e. metricsResolution or outlineResolution as appropriate. However, this value in points still needs to be converted into a value in pixels.

Computer display systems typically assume a 72 pixel per inch display. So, as each point is 1/72 inch, the horizontal and vertical size of each pixel is 1 point.

D.3.4.1 Vertical resolution

Each pixel in the graphics device containing the component is equivalent to a single point.

D.3.4.2 Horizontal resolution

The horizontal relationship depends on the characteristics of the graphics device. For a square pixel graphics device the 1 pixel = 1 point convention can be preserved.

However, for a graphics device whose pixel aspect ratio <HAVi method> the horizontal resolution is the pixel aspect ratio * 1 point.

Table D.3 : Pixel width for non-square pixel graphics devices

Graphics device resolution	Graphics device aspect ratio	Typographic pixel size width in points
720 x 576	4:3	128/117
	14:9	56/45
	16:9	512/351

An emulated graphics could be constructed with 14:9 aspect ratio. This could be used where text is required to be acceptable when viewed on either a 4:3 or 16:9 display. A possible example of this is illustrated in figure D.2.

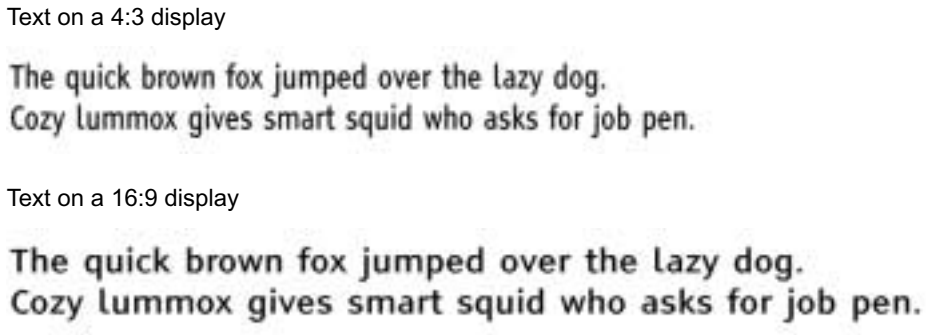


Figure D.2 : Example of 14:9 text on either 4:3 or 16:9 display

D.3.5 Rendering within limits

When typesetting for print, character extremities may extend beyond the nominal text flow area. However, print has margins so the edge of the text flow is not the technical limit to the area that can be printed. Component Insets can be used for the same purpose.

In "Low level rendering" the author is responsible for placing the text so that is not clipped.

In "High level rendering" the layout manager places the text within the inset rectangle, so that clipping does not occur if the object is sufficiently large. So, the "virtual margin" shall be defined as the greater of the inset value and the value computed using properties of the font (`xMin`, `yMin`, `xMax` and `yMax`) to ensure that all presented characters are completely rendered within the bounds of the object.

As stated previously, these parameters are defined in outline resolution units and so need to be converted to device pixels. Based on the principles described previously (see "Converting font metrics to display pixels") this can be achieved by using the following:

For `yMin and yMax`

Y_{points} = (fontsize x $Y_{outlineResolution}$) / outlineResolution

Y_{pixels} = roundupmag(Y_{points})

and for `xMin and xMax`

X_{points} = (fontsize x $X_{outlineResolution}$) / outlineResolution

X_{pixels} = roundupmag((X_{points} / pixel_aspect_ratio)

outlineResolution is extracted from the font. 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.3.5.1 Vertical limits

The origin of any character shall be at least `yMax` inside the top edge of the object and at least `yMin` inside its bottom edge. The distance from the edge of the component may be further increased by any margin. Assuming that all characters in a line of text share a common baseline then, **regardless of vAlign** the number of lines of text that may be presented within a text object is:

num_lines = floor((object_height - (`yMin + yMax`)) / linespace) + 1

Where object_height is the height of the component less any margin.

All values are in pixels. The variable linespace is an attribute of the object that defines the space between the baselines of consecutive lines of text. The function floor(A) rounds A to the first integral number less than or equal to A.

Note: Linespace is defined in units of points but as described previously (see "Converting font metrics to display pixels").

- When `vAlign` = `TOP_ALIGN` the baseline of the top most line shall be `yMax` inside the "virtual margin" and each following line shall be spaced according to the value of linespace.

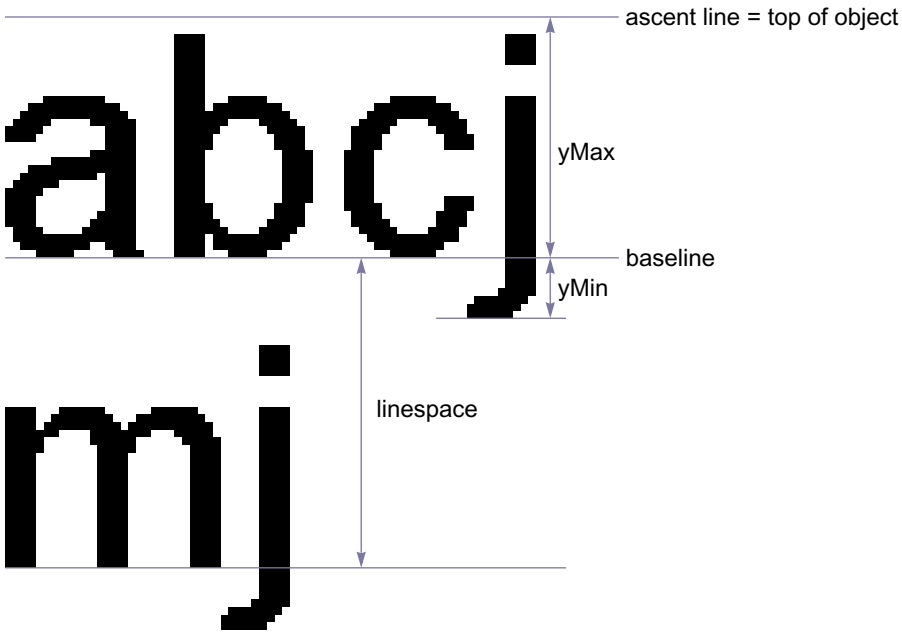


Figure D.3 : Vertical measures

- When `vAlign` = `BOTTOM_ALIGN` the origin of the bottom most line shall be `yMin` inside the "virtual margin" of the object and each previous line shall be spaced according to the value of linespace.
- For centring the midpoint of each line shall be the midpoint of the distance between the relevant "virtual margins"

D.3.5.2 Horizontal limits

The number of characters that may be rendered on a line is not simply dependent upon the width of the object and the horizontal escapement for each character, but also needs to consider that the rendering of the first character in a line may extend to the left of its origin. Thus, **regardless of** `hAlign` the space available for rendering a line of text within an object is

available_width = object_width - `xMin`

All values are in pixels. available_width may then be used with the "logical" text width rules to determine text flow.

- When `hAlign` = LEFT_ALIGN the origin of the left most character shall be `xMin` inside the left edge of the "virtual margin".
- When `hAlign` = RIGHT_ALIGN the origin of the right most character shall be as necessary to ensure that it is completely visible when rendered.

D.3.6 "logical" text width rules

This clause applies to both "Low level rendering" and "High level rendering". Its purpose is to ensure that text will flow predictably on different receivers and authoring stations, regardless of the quality of the character rendering, a set of "logical" text width rules are defined here.

NOTE: I.e. lines and words will break at the same character position.

These rules are a simplification of the rules that might be applied in a typographic rendering system. The objective of these simplifications is to reduce the receiver complexity required to ensure exact correlation of text flow behaviour.

The calculation of "logical" text width is based on "physical" font data. This data provides a description of the font at a very high resolution, abstracted from any actual rendering system. Consequently, the calculation of the "logical" width of a string of characters involves, computing their width at this high resolution and then converting to units appropriate to the rendering system, e.g. device pixels, before making decisions about text flow (see "Converting font metrics to display pixels").

In the case of "Low level rendering" it defines the internal computation performed by the AWT routines that measure the width of text:

- `java.awt.FontMetrics.charWidth`
- `java.awt.FontMetrics.stringWidth`
- `java.awt.FontMetrics.bytesWidth`

Due to the rounding processes within the calculations invoking these methods on subsets of a string may not yield the same total result as invoking the methods on the complete string. In particular the total of the values returned by `java.awt.FontMetrics.getWidths` may be different from the value returned by `java.awt.FontMetrics.stringWidth` for the same string.

In the case of "High level rendering" it defines the computations that the layout manager uses in the following cases:

- to determine when to wrap lines of text within an object
- to determine which tab stops text has passed when implementing tab characters

D.3.6.1 Computing "logical" text width

The key parameters when calculating the width of a string of N characters are:

- text font size
- charSetWidth
- The metricsResolution
- Any kerning adjustment

D.3.6.1.1 Font sizes

Font sizes are expressed as the size of an "Em" in units of "points".

NOTE 1: Broadly speaking an Em is the minimum distance between the baselines of consecutive lines of text in the given font. If text is 48 point then the Em at that size is 48 points.

NOTE 2: The point is an archaic typographical unit. Traditionally there were 72.27 points to an inch. Computerised systems now use 72 points per inch for simplicity.

D.3.6.1.2 Character widths

The font definition gives the width of each character relative to the size of an Em in metricsResolution units.

NOTE: If metrics are specified in 1/1000ths of an Em a character with a width of 0.6 Em will have a set width of 600.

D.3.6.1.3 Kerning

For certain character combinations (a "kerning pair") a kerning adjustment may also be provided. Typically kerning reduces character spacing for pairs such as AV instead of A V, these provide a signed adjustment to the nominal charSetWidth of the first character.

Like charSetWidth kerning adjustments are in terms of metricsResolution units.

Kerning adjustments only apply between non whitespace characters, not between the start of a line of text and the edge of the text object. If justification is being used, only whitespace between words may be adjusted.

D.3.6.1.4 Tracking

Tracking allows for an expansion/condensation of the character spacing for all of the characters in a text object including whitespace.

D.3.6.2 Logical text width

The equation below shows how the width of a string of N characters is computed.

$$\text{logical width of N characters}_{\text{points}} =$$

$$\text{div}((N-1)\times \text{track},256) + \text{div}\left(\text{fontsize}\times\left(\sum_{1}^{N}\text{charSetWidth[i]} + \sum_{1}^{N-1}\text{kern[i,i+1]}\right),\text{metricsResolution}\right)$$

$$\text{logical width of N characters}_{\text{pixels}} = \text{div}(\text{logical width of N characters}_{\text{points}}\times H, W)$$

Where in div(A, B):

- B is unsigned and A is signed

  and

- div(A, B) = ceil(A / B)

Where '/' is a rational divide and ceil(A) is the first integral number greater than or equal to A. So, the calculations round up when reducing precision and tend to over estimate the width of text.

Where H and W are respectively the height and width returned by `org.havi.ui.HScreenConfiguration.getPixelAspectRatio()`.

D.3.7 Line breaking

The text layout manager shall wrap text within the text object. The behaviour is equivalent to identifying (based on the "logical" length) the first word that won't fit completely within the text object and replacing the space character that precedes the word with a Carriage Return character. I.e. line breaks are only inserted where there are space characters, this implies the receiver does not have to apply word hyphenation rules.

If a single word is bigger than the text object the text layout manager shall truncate text just before the last character that won't fit completely within the text box.

D.3.7.1 Truncation of text

High Level Text rendering shall not result in the presentation of partial characters.

If an object is too small to display all of the text that it contains the text shall be truncated so that only complete characters are displayed, the notifyTextOverflow method will be called so that the application can alert the viewer or take other remedial action.

If the object is not tall enough to be able to display all of the lines of text loaded to it then the rendering shall only display lines whose height can be completely displayed.

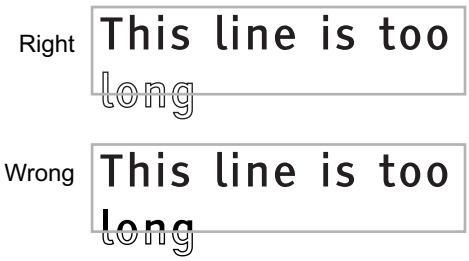


Figure D.4 : Truncation at vertical 'end' edge

The examples illustrate the behaviour where `hAlign` is LEFT_ALIGN and vAlign is TOP_ALIGN. For other values the handling should be based on the same principles.

D.3.8 Tabulation

In left aligned text tab stops are defined by default horizontally every 56 points from the left edge of the text box.

'Horizontal Tabulation' advances the origin of the next character to be rendered to the next tab stop in the direction that the text is currently flowing (the character repertoire in Table E.1 only requires left to right text).

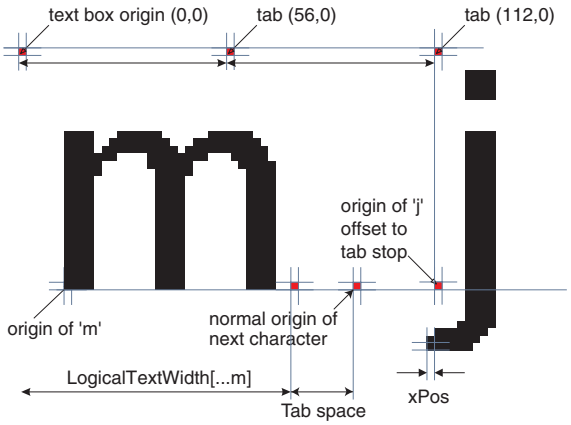


Figure D.5 : Effect of horizontal tabulation

- Tab characters only have meaning in left aligned text. If the text is right aligned, centred or justified then tab character shall be treated as a space character.
- A tab logically advances the rendering of the text by at least the width of a space character. If the normal origin of the next character to be rendered after the tab character is after a tab stop, a tab character will advance the rendering to the subsequent tab stop.
- The tab stops are at regular intervals from the left edge of the object and are not affected by the `xMin` offset to the origin of the first character.

D.3.9 Placing runs of characters & words

A run of characters starts from a well defined point:

- The start edge of the text object (see D.3.5 on page 204)
- A tab stop

After this origin the fine positioning of character cells and the gaps between words is not fully specified.

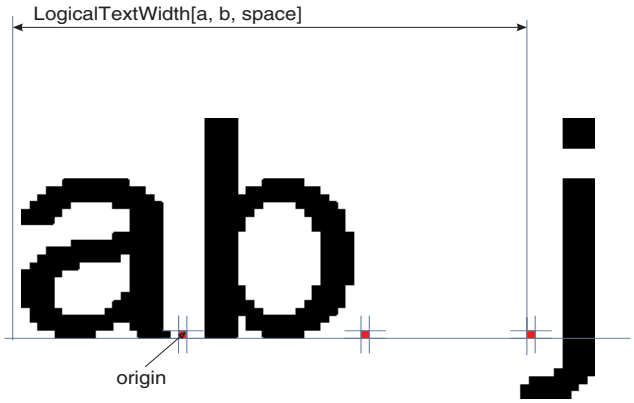


Figure D.6 : Calculation of character placement

However, the following rendering requirements shall be observed to ensure that a minimum acceptable level of text legibility is achieved:

- The spacing between any pair of non whitespace characters should be consistent wherever that pair of characters is displayed.
- At the default character spacing no two non-whitespace characters should "appear" to touch, except for special cases such as where ligatures dipthongs etc. are being used. The definition of the underlying font may make this requirement impossible.
- The physical rendering of a run of text as determined by the "logical" rules shall be achieved completely within the space used for the "logical" calculation.
- No partially rendered characters shall be presented.

D.3.10 Control of text flow

See the definition of the DVB text layout manager.

D.4 Text mark-up

This clause on text mark-up applies to "High level rendering".

D.4.1 White Space Characters

Certain non-printing characters have special meaning. These are identified in Table D.4.

Table D.4 : Special characters (Sheet 1 of 2)

UTF8 Value(s)	Name	Meaning
0x09	Tab	See “Tabulation” on page 208
0x0D	Carriage Return	Causes the text flow to break. The origin for the next character to be rendered moves to a new baseline "linespace" below that just rendered. The horizontal position of the next line will depend on the `hAlign`.

Table D.4 : Special characters (Sheet 2 of 2)

UTF8 Value(s)	Name	Meaning
0x20	Space	Spaces text by the width defined for the space character. When a text object has the TextWrapping attribute set to "true" lines may be broken at a space. See D.3.7, "Line breaking" on page 207.
0xC2 0xA0	Non-breaking space	Identical spacing characteristics to 0x20 but is not seen as word boundary for deciding a position to break a line of text. (0xC2A0 is the UTF-8 representation of 0x00A0)

D.4.2 Marker characters

The codes 0x1C to 0x1F are zero width, non-spacing, non-printing characters available for use by authors as markers in text objects, i.e. when using string operations.

D.4.3 Non-printing characters

Certain characters (or character sequences) have no immediate visual representation.

These include:

- 0x1C to 0x1F marker characters (see D.4.2 on page 210)
- Format control mark-up (see D.4.4 on page 210)
- other characters not recognised by the receiver

When presenting text that includes these characters the character placement shall be as if the non-printing characters were eliminated from the text before rendering. In particular, the character spacing and inter character kerning shall be computed as if the non-printing characters were not present.

D.4.4 Format Control Mark-up

Within text objects mark-up codes can be used to control the presentation of text. The sequence in table D.5 marks the start of some marked-up text. For each "start of mark-up" a corresponding "end of mark-up" is defined. The byte sequence for the "end of mark-up" is illustrated in Table D.6. The minimum number of supported mark-up instances, where each instance is a start and end mark-up pair, is 256.

Table D.5 : General format for start of text mark-up

	bits	value	note
`start_of_markup`	8	0x1B	Escape
`markup_start_identifier`	8	0x40-0x5E	'@' to '^'
`parameters_length`	8	N	
`for( i=0; i<N; i++ ) {`			
`  parameter_byte`	8	0x00…0xFF	
`}`			

Table D.6 : General format for end of text mark-up

	bits	value	note
`end_of_markup`	8	0x1B	Escape
`markup_end_identifier`	8	0x60-0x7E	'`' to '~'

Table D.7 : Text object mark-up codes

Min. Nesting	start mark-up	end mark-up	description
	0x1B 0x42 0x00	0x1B 0x62	Applies 'bold' style to the text enclosed (note 1)
16	0x1B 0x43 0x04 0xrr 0xgg 0xbb 0xtt	0x1B 0x63	Applies colour to the text enclosed. 0xrr specifies the red intensity, 0xgg the green, 0xbb the blue and 0xtt the transparency.
NOTE 1: Not supported in this profile			

D.4.5 Future compatibility

Compatible extensions to the set of mark-up codes may be defined in future profiles. For each the markup_end_identifier will be 32 (0x20) greater than the markup_start_identifier. Engines should ignore unrecognised mark-up and should 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".

Table E.1 : Extended Latin set (Sheet 1 of 9)

Code	Unicode 2.0 Character name
0x0020	SPACE
0x0021	EXCLAMATION MARK
0x0022	QUOTATION MARK
0x0023	NUMBER SIGN
0x0024	DOLLAR SIGN
0x0025	PERCENT SIGN
0x0026	AMPERSAND
0x0027	APOSTROPHE
0x0028	LEFT PARENTHESIS
0x0029	RIGHT PARENTHESIS
0x002A	ASTERISK
0x002B	PLUS SIGN
0x002C	COMMA
0x002D	HYPHEN-MINUS
0x002E	FULL STOP
0x002F	SOLIDUS
0x0030	DIGIT ZERO
0x0031	DIGIT ONE
0x0032	DIGIT TWO
0x0033	DIGIT THREE
0x0034	DIGIT FOUR
0x0035	DIGIT FIVE
0x0036	DIGIT SIX
0x0037	DIGIT SEVEN
0x0038	DIGIT EIGHT
0x0039	DIGIT NINE
0x003A	COLON
0x003B	SEMICOLON
0x003C	LESS-THAN SIGN
0x003D	EQUALS SIGN
0x003E	GREATER-THAN SIGN
0x003F	QUESTION MARK
0x0040	COMMERCIAL AT
0x0041	LATIN CAPITAL LETTER A
0x0042	LATIN CAPITAL LETTER B

Table E.1 : Extended Latin set (Sheet 2 of 9)

Code	Unicode 2.0 Character name
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
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

Table E.1 : Extended Latin set (Sheet 3 of 9)

Code	Unicode 2.0 Character name
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
0x00CA	LATIN CAPITAL LETTER E WITH CIRCUMFLEX
0x00CB	LATIN CAPITAL LETTER E WITH DIAERESIS
0x00CC	LATIN CAPITAL LETTER I WITH GRAVE

Table E.1 : Extended Latin set (Sheet 4 of 9)

Code	Unicode 2.0 Character name
0x00CD	LATIN CAPITAL LETTER I WITH ACUTE
0x00CE	LATIN CAPITAL LETTER I WITH CIRCUMFLEX
0x00CF	LATIN CAPITAL LETTER I WITH DIAERESIS
0x00D0	LATIN CAPITAL LETTER ETH
0x00D1	LATIN CAPITAL LETTER N WITH TILDE
0x00D2	LATIN CAPITAL LETTER O WITH GRAVE
0x00D3	LATIN CAPITAL LETTER O WITH ACUTE
0x00D4	LATIN CAPITAL LETTER O WITH CIRCUMFLEX
0x00D5	LATIN CAPITAL LETTER O WITH TILDE
0x00D6	LATIN CAPITAL LETTER O WITH DIAERESIS
0x00D7	MULTIPLICATION SIGN
0x00D8	LATIN CAPITAL LETTER O WITH STROKE
0x00D9	LATIN CAPITAL LETTER U WITH GRAVE
0x00DA	LATIN CAPITAL LETTER U WITH ACUTE
0x00DB	LATIN CAPITAL LETTER U WITH CIRCUMFLEX
0x00DC	LATIN CAPITAL LETTER U WITH DIAERESIS
0x00DD	LATIN CAPITAL LETTER Y WITH ACUTE
0x00DE	LATIN CAPITAL LETTER THORN
0x00DF	LATIN SMALL LETTER SHARP S
0x00E0	LATIN SMALL LETTER A WITH GRAVE
0x00E1	LATIN SMALL LETTER A WITH ACUTE
0x00E2	LATIN SMALL LETTER A WITH CIRCUMFLEX
0x00E3	LATIN SMALL LETTER A WITH TILDE
0x00E4	LATIN SMALL LETTER A WITH DIAERESIS
0x00E5	LATIN SMALL LETTER A WITH RING ABOVE
0x00E6	LATIN SMALL LETTER AE
0x00E7	LATIN SMALL LETTER C WITH CEDILLA
0x00E8	LATIN SMALL LETTER E WITH GRAVE
0x00E9	LATIN SMALL LETTER E WITH ACUTE
0x00EA	LATIN SMALL LETTER E WITH CIRCUMFLEX
0x00EB	LATIN SMALL LETTER E WITH DIAERESIS
0x00EC	LATIN SMALL LETTER I WITH GRAVE
0x00ED	LATIN SMALL LETTER I WITH ACUTE
0x00EE	LATIN SMALL LETTER I WITH CIRCUMFLEX
0x00EF	LATIN SMALL LETTER I WITH DIAERESIS

Table E.1 : Extended Latin set (Sheet 5 of 9)

Code	Unicode 2.0 Character name
0x00F0	LATIN SMALL LETTER ETH
0x00F1	LATIN SMALL LETTER N WITH TILDE
0x00F2	LATIN SMALL LETTER O WITH GRAVE
0x00F3	LATIN SMALL LETTER O WITH ACUTE
0x00F4	LATIN SMALL LETTER O WITH CIRCUMFLEX
0x00F5	LATIN SMALL LETTER O WITH TILDE
0x00F6	LATIN SMALL LETTER O WITH DIAERESIS
0x00F7	DIVISION SIGN
0x00F8	LATIN SMALL LETTER O WITH STROKE
0x00F9	LATIN SMALL LETTER U WITH GRAVE
0x00FA	LATIN SMALL LETTER U WITH ACUTE
0x00FB	LATIN SMALL LETTER U WITH CIRCUMFLEX
0x00FC	LATIN SMALL LETTER U WITH DIAERESIS
0x00FD	LATIN SMALL LETTER Y WITH ACUTE
0x00FE	LATIN SMALL LETTER THORN
0x00FF	LATIN SMALL LETTER Y WITH DIAERESIS
0x0100	LATIN CAPITAL LETTER A WITH MACRON
0x0101	LATIN SMALL LETTER A WITH MACRON
0x0102	LATIN CAPITAL LETTER A WITH BREVE
0x0103	LATIN SMALL LETTER A WITH BREVE
0x0104	LATIN CAPITAL LETTER A WITH OGONEK
0x0105	LATIN SMALL LETTER A WITH OGONEK
0x0106	LATIN CAPITAL LETTER C WITH ACUTE
0x0107	LATIN SMALL LETTER C WITH ACUTE
0x0108	LATIN CAPITAL LETTER C WITH CIRCUMFLEX
0x0109	LATIN SMALL LETTER C WITH CIRCUMFLEX
0x010A	LATIN CAPITAL LETTER C WITH DOT ABOVE
0x010B	LATIN SMALL LETTER C WITH DOT ABOVE
0x010C	LATIN CAPITAL LETTER C WITH CARON
0x010D	LATIN SMALL LETTER C WITH CARON
0x010E	LATIN CAPITAL LETTER D WITH CARON
0x010F	LATIN SMALL LETTER D WITH CARON
0x0110	LATIN CAPITAL LETTER D WITH STROKE
0x0111	LATIN SMALL LETTER D WITH STROKE
0x0112	LATIN CAPITAL LETTER E WITH MACRON

Table E.1 : Extended Latin set (Sheet 6 of 9)

Code	Unicode 2.0 Character name
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
0x0135	LATIN SMALL LETTER J WITH CIRCUMFLEX
0x0136	LATIN CAPITAL LETTER K WITH CEDILLA

Table E.1 : Extended Latin set (Sheet 7 of 9)

Code	Unicode 2.0 Character name
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
0x015F	LATIN SMALL LETTER S WITH CEDILLA
0x0160	LATIN CAPITAL LETTER S WITH CARON

Table E.1 : Extended Latin set (Sheet 8 of 9)

Code	Unicode 2.0 Character name
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
0x1E84	LATIN CAPITAL LETTER W WITH DIAERESIS

Table E.1 : Extended Latin set (Sheet 9 of 9)

Code	Unicode 2.0 Character name
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.

Annex G (normative): Minimum Platform Capabilities

G.1 Graphics

In the area of graphics capability the following requirements are made on MHP terminals:

G.1.1 Device capabilities

- The number of applications concurrently using the display is not limited. However, the MHP terminal is not required to support overlapping HScenes.
- The MHP terminal shall implement at least one HGraphicsDevice which shall be full screen.
- The MHP terminal shall implement at least one HBackgroundDevice. These are always full screen.
- The MHP terminal shall implement at least one HVideoDevice which is always capable of being configured to be full screen.
- The minimum set of required device resolutions that MHP terminals shall support is illustrated in figure G.1. Specifically these are:
  - HBackgroundDevice resolution of 720x576
  - HVideoDevice resolution of 720x576
  - HGraphicsDevice resolution of 720x576

  These shall be supported for display aspect ratios of 4:3 and 16:9.

  Optionally MHP terminals may also support square pixel HGraphicsDevice resolutions of 768x576 and 1024x576 for 4:3 and 16:9 displays respectively.

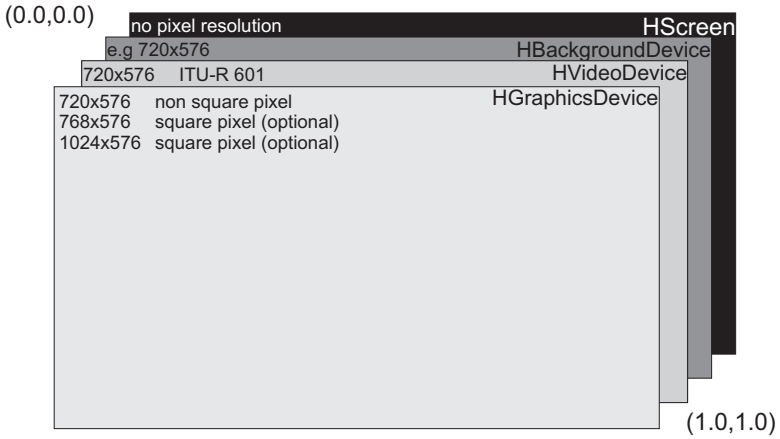


Figure G.1 : Required device resolutions

G.1.2 Video presentation capabilities

- The following set of standard decoder format conversions shall be supported by all MHP terminals:

  DFC_PROCESSING_CCO

  DFC_PROCESSING_FULL

  DFC_PROCESSING_LB_16_9

  DFC_PROCESSING_PAN_SCAN

  The following modes are optional:

  DFC_PROCESSING_LB_14_9

  DFC_PROCESSING_LB_2_21_1_ON_16_9

DFC_PROCESSING_LB_2_21_1_ON_4_3

- Either full screen or 1/4 screen where the video area is fully on screen.

G.1.3 Image processing capabilities

- All DVBGraphics objects shall support SRC and CLEAR and SRC_OVER.
  When SRC_OVER is used with DVBGraphics objects which a sample model of the type TYPE_BASE a perfect result is only produced with alpha values of 0 and 1. Alpha values other than 0 and 1 can be approximated. DVB-Graphics object with a type of TYPE_ADVANCED will produce a result as expected but those SRC_OVER operations are likely to be slow.

- DVBGraphics object created from a DVBBufferedImage with the type TYPE_ADVANCED shall perform SRC_OVER operations without approximations of the compositing rule.

- Where the DVBGraphics object is created from a DVBBufferedImage with the type TYPE_ADVANCED full alpha blending shall be implemented when the composition mode is SRC_OVER.

G.1.4 Alpha capabilities

For any draw operations directly into the HGraphicsDevice the following rules shall be applied for the precision of implementation of alpha:

• MHP terminals are required to implement at least 3 levels of transparency 0% (opaque), 30% and 100% (completely transparent). 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%.

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 (Sheet 1 of 2)

Transparency		Additional Grey Levels (R=G=B)	Red	Green	Blue	Number of colours
0%	0x00	42, 85, 170,212	0, 63, 127, 191, 255	0, 31, 63, 95, 127, 159, 191, 223, 255	0, 127, 255	139
30%	0x2C (note 1)	--	0, 85, 170, 255	0, 51, 102, 153, 204, 255	0, 255	48

Table G.1 : Palette construction rules (Sheet 2 of 2)

Transparency		Additional Grey Levels (R=G=B)	Red	Green	Blue	Number of colours
100%	0xFF	--	--	--	--	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.

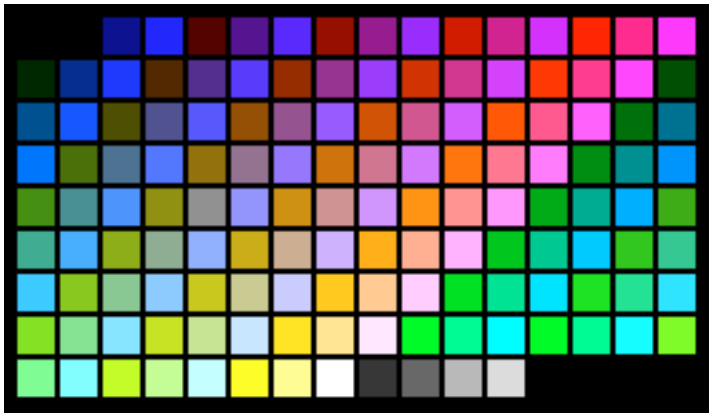


Figure G.2 : Opaque CLUT

G.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.

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

G.4.1 The built-in font

At least the RNIB/DTG font "Tiresias" shall be provided.

The font shall be able to present at least the sizes listed in G.2 and the weight ("PLAIN").

Table G.2 : Minimum set of sizes

Size (points)	TV lines (note 1) over 'Cap-V'	Informative Name
36	24	Heading / Large subtitle
31	21	Subtitle
26	18	Body (note 2)

Table G.2 : Minimum set of sizes

Size (points)	TV lines (note 1) over 'Cap-V'	Informative Name
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" is 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.5 Input events

Table G.3 : Minimum set of input events

Input event
VK_0 to VK_9
VK_UP
VK_DOWN
VK_LEFT
VK_RIGHT
VK_ENTER
VK_TELETEXT
VK_COLORED_KEY_0
VK_COLORED_KEY_1
VK_COLORED_KEY_2
VK_COLORED_KEY_3

G.6 Other resources

Table G.4 :

Feature	Specification
gamma correction in the receiver	none
HAVi mattes	Platforms are not required to implement the functionality of mattes in HAVi. Non-implementation should be implemented as specified by HAVi.
Overlapping applications	MHP terminals are not required to support overlapping top level UI containers (e.g. HScenes where DVB-J applications are concerned).
AIT section filtering	The implementation is not required to dedicate more than one section filter to monitoring the AIT.
Key lengths	Receivers shall support certificate key length up to 4096 bits.

Annex H (informative): Extensions

Private protocols and possibly APIs are not precluded and are outside of the scope of the MHP specification.

Annex I (normative): DVB-J fundamental classes

Package
org.dvb.lang

Class Summary	
Classes	
DVBClassLoader	This class loader is used to load classes and resources from a search path of locators referring to locations where Java class files may be stored.

org.dvb.lang
DVBClassLoader

Syntax

```
public class DVBClassLoader extends java.security.SecureClassLoader

java.lang.Object
  |
  +--java.lang.ClassLoader
        |
        +--java.security.SecureClassLoader
              |
              +--org.dvb.lang.DVBClassLoader
```

Description

This class loader is used to load classes and resources from a search path of locators 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 locators specified when the DVBClassLoader was created.

Constructors

DVBClassLoader(Locator[])

```
public DVBClassLoader(org.davic.net.Locator[] locators)
```

Constructs a new DVBClassLoader for the given locators. The locators 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:

`locators` - the locators 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(Locator[], ClassLoader)

```
public DVBClassLoader(org.davic.net.Locator[] locators, java.lang.ClassLoader parent)
```

Constructs a new DVBClassLoader for the given locators. The locators will be searched in the order specificied 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:

`locators` - the locators from which to load classes and resources

`parent` - the parent classloader for delegation

Throws:
`SecurityException` - if a security manager exists and its `checkCreateClassLoader` method doesn't allow creation of a class loader.

See Also:
`SecurityManager`

Methods

---

findClass(String)

```
public java.lang.Class findClass(java.lang.String name)
```

Finds and loads the class with the specified name from the locator search path. Any locators 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(Locator[])

```
public static DVBClassLoader newInstance(org.davic.net.Locator[] locators)
```

Creates a new instance of DVBClassLoader for the specified locators. 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:
`locators` - the locators to search for classes and resources.

Returns:
the resulting class loader

---

newInstance(Locator[], ClassLoader)

```
public static DVBClassLoader newInstance(org.davic.net.Locator[] locators,
            java.lang.ClassLoader parent)
```

Creates a new instance of DVBClassLoader for the specified locators. 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:
`locators` - the locators to search for classes and resources.

`parent` - the parent class loader for delegation.

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 an application with different ways to catch user events. These are:

- through the standard `java.awt` mechanism,
- through the standard `java.awt` mechanism, but for some events to be exclusively accessed by the application,
- through a mechanism defined by this API,
- through the mechanism defined by this API, but for some events to be exclusively accessed by the application.

The last two solutions could be used by non-graphical applications in order to receive events that are coming from the user. It could also be used by an invisible application if it wants to be presented when a specific key is pressed.

If an application wants to have exclusive access to some events and to manage them through the java.awt then it must use this API so that it can be aware of the fact that it has lost or gained access to these events. One must notice that an application based on awt event mechanism will receive events only if it is focused.

The diagram below shows how an event is dispatched depending on how it was requested:

```
User Event
    |
    |
    . Is it in a repository used by the Event Manager?
    |    |
    yes  no
    |    |
    |    --> send the event to the awt application which is focused.
    |
    . Is it an exclusive event?
    |    |
    yes  no
    |    |
    |    --> send the event to the applications which do not use the awt
    |        event mechanism and have asked for this event.
    |
    . Was it acquired by an awt application?
    |    |
    yes  no
    |    |
    |    .-> send the event to the application which does not use the
    |        awt event mechanism and has asked for an exclusive access
    |        to this event.
    |
    . -----> send the event to the application if it is focused.
```

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.UEC_OK) ;
           em.addUserListener ((UserEventListener)this, (ResourceClient)this, repository) ;
           em.addResourceStatusEventListener (this) ;
       }
       /**
        * methods defined by the UserEventListener interface.
        */
       public void UserEventReceived (UserEvent e) {
       }
       /**
        * Methods defined by the ResourceClient interface.
        */
       /**
        * In the case a cooperative application asks for an user event
        * exclusively used by me.
        */
       public boolean requestRelease(ResourceProxy proxy, Object requestData) {
           String name ;
           // let's retrieve the name of the repository, that I have created, and
           // which contains the user event that the other application asks for.
           name = (RepositoryDescriptor)proxy.getName () ;
           if ((name.compareTo ("R1") == 0) & (myStatus == ...)) {
               // Ok I release this event.
               return true ;
           } else {
               // No I need this event, sorry !
               return false ;
```

```
        }
    }
    public void release (ResourceProxy proxy) {
        ...
    }
    public void notifyRelease (ResourceProxy proxy) {
        ...
    }
    public void statusChanged (ResourceStatusEvent event) {
        ...
    }
}
```

Package
org.dvb.event

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 contains all the user events defined in the UserEvent class.
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.
UserEventRepository	The application will use this class to define the events that it wants to receive.

org.dvb.event
EventManager

Syntax
```
public final class EventManager implements org.davic.resources.ResourceServer

java.lang.Object
  |
  +--org.dvb.event.EventManager
```

All Implemented Interfaces:
org.davic.resources.ResourceServer

Description
The event manager allows an application to receive events coming from the user. These events can be sent exclusively to an application or can be shared between applications. The Event Manager allows also the application to ask for exclusive access to some events, these events being received either from the standard java.awt event mechanism or by the mechanism defined in this package. The EventManager is a singleton.

Methods

---

addExclusiveAccessToAWTEvent(ResourceClient, UserEventRepository)

```
public boolean addExclusiveAccessToAWTEvent(org.davic.resources.ResourceClient client,
            UserEventRepository userEvents)
```

An application should use this method to express its intend to have exclusive access to some events, but for these events to be received through the java.awt mechanism. The events the application wishes to receive are defined by the means of the UserEventRepository class. This repository is resolved at the time when this method call is made and adding or removing events from the repository after this method call doesn't affect the subscription to those events. An exclusive event will be sent to the application if this latest is focused.

Parameters:

`client` - resource client.

`userEvents` - the user events the application wants to be inform of.

Returns:

true if the events defined in the repository have been acquired, false otherwise.

Throws:

`IllegalArgumentException` - if the client argument is set to null.

---

addResourceStatusEventListener(ResourceStatusListener)

```
public void addResourceStatusEventListener(org.davic.resources.ResourceStatusListener
            listener)
```

Adds the specified resource status listener so that an application can be aware of any changes regarding exclusive access to some events.

Specified By:

org.davic.resources.ResourceServer.addResourceStatusEventListener(org.davic.resources.ResourceStatusListener) in interface org.davic.resources.ResourceServer

Parameters:

`listener` - the resource status listener.

---

addUserEventListener(UserEventListener, ResourceClient, UserEventRepository)

```
public boolean addUserEventListener(UserEventListener listener,
            org.davic.resources.ResourceClient client, UserEventRepository userEvents)
```

Adds the specified listener to receive events coming from the user in an exclusive manner. The events the application wishes to receive are defined by the means of the UserEventRepository class. This repository is resolved at the time when this method call is made and adding or removing events from the repository after this method call doesn't affect the subscription to those events. The ResourceClient parameter indicates that the application wants to have an exclusive access to the user event defined in the repository. A null value for this parameter means that events in the repository will not be accessed exclusively.

Parameters:

`listener` - the listener to receive the user events.

`client` - resource client.

`userEvents` - a class which contains the user events it wants to be informed of.

Returns:

true if the events defined in the repository have been acquired, false otherwise.

Throws:

`IllegalArgumentException` - if the client argument is set to null.

---

addUserEventListener(UserEventListener, UserEventRepository)

```
public void addUserEventListener(UserEventListener listener,
            UserEventRepository userEvents)
```

Adds the specified listener to receive events coming from the user. The events the application wishes to receive are defined by the means of the UserEventRepository class. This repository is resolved at the time when this method call is made and adding or removing events from the repository after this method call doesn't affect the subscription to those events.

Parameters:

`listener` - the listener to receive the user events.

`userEvents` - a class which contains the user events it wants to be informed of.

---

getInstance()

```
public static EventManager getInstance()
```

This method returns the sole instance of the EventManager class. The EventManager class is a singleton.

Returns:

the instance of the EventManager.

---

removeExclusiveAccessToAWTEvent(ResourceClient)

```
public void removeExclusiveAccessToAWTEvent(org.davic.resources.ResourceClient client)
```

The application should use this method to release its exclusive access to user events defined by the means of the addExclusiveAccessToAWTEvent method.

Parameters:

`client` - the client that is no longer interested in events previously registered.

---

removeResourceStatusEventListener(ResourceStatusListener)

```
public void removeResourceStatusEventListener(org.davic.resources.ResourceStatusListener
            listener)
```

Removes the specified resource status listener.

Specified By:

org.davic.resources.ResourceServer.removeResourceStatusEventListener(org.davic.resources.ResourceStatusListener) in interface org.davic.resources.ResourceServer

Parameters:

`listener` - the listener to remove.

---

removeUserEventListener(UserEventListener)

```
public void removeUserEventListener(UserEventListener listener)
```

Removes the specified listener so that it will no longer receives user events. If it is appropriate (i.e the application has asked for an exclusive access), the exclusive access is lost.

Parameters:

`listener` - the user event listener.

org.dvb.event
OverallRepository

Syntax
```
public class OverallRepository extends UserEventRepository

java.lang.Object
  |
  +--UserEventRepository
        |
        +--org.dvb.event.OverallRepository
```

Description
This class defines a repository which contains all the user events defined in the UserEvent class. For example, this pre-defined repository could be used by an application, which requires a pin code from the user, in order to prevent another applications from receiving events.

See Also:
UserEvent

Constructors

OverallRepository()

```
public OverallRepository()
```

org.dvb.event
RepositoryDescriptor

Syntax
```
public class RepositoryDescriptor implements org.davic.resources.ResourceProxy

java.lang.Object
  |
  +--org.dvb.event.RepositoryDescriptor
```

All Implemented Interfaces:
org.davic.resources.ResourceProxy

Description
An instance of this class will be sent to clients of the DVB event API to notify them (through the interface org.davic.resources.ResourceClient) when they are about to lose, or have lost, access to an event source. This object can be used by the application to get the name of the repository from which it will no longer be able to receive events.

Methods

---

getClient()

```
public org.davic.resources.ResourceClient getClient()
```

Specified By:
org.davic.resources.ResourceProxy.getClient() in interface org.davic.resources.ResourceProxy

---

getName()

```
public java.lang.String getName()
```

Returns the name of the repository to which the lost user event belongs.

Returns:
String the name of the repository.

org.dvb.event
UserEvent

Syntax
```
public class UserEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.event.UserEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
Represents a user event. A user event is defined by a family, a type and an code.

Values for type are the constants KEY_PRESSED and KEY_RELEASED defined in java.awt.event.KeyEvent. Values for code are the constants whose names begin with "VK_" defined in that class and in the org.havi.ui.event package.

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, int, int)

```
public UserEvent(java.lang.Object source, int family, int type, int code)
```

Constructor for a new UserEvent object.

Parameters:

`source` - the `EventManager` which is the source of the event

`family` - the event family.

`type` - the event type

`code` - the event code.

Methods

getCode()

```
public int getCode()
```

Returns the event code.

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.

getType()

```
public int getType()
```

Returns the event type. Could be KEY_PRESSED, KEY_RELEASED

Returns:
an int representing the event type.

org.dvb.event
UserEventListener

Syntax
```
public interface UserEventListener extends java.util.EventListener
```

All Superinterfaces:
java.util.EventListener

Description
The listener interface for receiving user inputs.

Methods

UserEventReceived(UserEvent)

```
public void UserEventReceived(UserEvent e)
```

org.dvb.event
UserEventRepository

Syntax
```
public class UserEventRepository

java.lang.Object
  |
  +--org.dvb.event.UserEventRepository
```

Direct Known Subclasses:
OverallRepository

Description
The application will use this class to define the events that it wants to receive. Events that are able to be put in the repository are defined in the UserEvent class.

See Also:
UserEvent

Constructors

UserEventRepository(String)

```
public UserEventRepository(java.lang.String name)
```

The method to construct a new UserEventRepository.

Parameters:

`name` - the name of the repository.

Methods

---

addAllArrowKeys()

```
public void addAllArrowKeys()
```

Adds the key codes for the arrow keys (VK_LEFT, VK_RIGHT, VK_UP, VK_DOWN). Any key codes already in the repository will not be added again.

---

addAllColourKeys()

```
public void addAllColourKeys()
```

Adds the key codes for the colour keys (VK_COLORED_KEY_0, VK_COLORED_KEY_1, VK_COLORED_KEY_2, VK_COLORED_KEY_3). Any key codes already in the repository will not be added again.

addAllNumericKeys()

```
public void addAllNumericKeys()
```

Adds the key codes for the numeric keys (VK_0, VK_1, VK_2, VK_3, VK_4, VK_5, VK_6, VK_7, VK_8, VK_9). Any key codes already in the repository will not be added again.

addKey(int)

```
public void addKey(int keycode)
```

A shortcut to create a new key event type entry in the repository. If a key is already in the repository, this method has no effect.

Parameters:

`int` - the key code.

addUserEvent(UserEvent)

```
public void addUserEvent(UserEvent event)
```

Creates a new user event.

Parameters:

`event` - the user event to be added in the repository.

getUserEvent()

```
public UserEvent[] getUserEvent()
```

Returns the list of the user events that are in the repository.

Returns:

an array which contains the user events that are in the repository.

removeAllArrowKeys()

```
public void removeAllArrowKeys()
```

Removes the key codes for the arrow keys (VK_LEFT, VK_RIGHT, VK_UP, VK_DOWN). Key codes from this set which are not present in the repository will be ignored.

removeAllColourKeys()

```
public void removeAllColourKeys()
```

Removes the key codes for the colour keys (VK_COLORED_KEY_0, VK_COLORED_KEY_1, VK_COLORED_KEY_2, VK_COLORED_KEY_3). Key codes from this set which are not present in the repository will be ignored.

removeAllNumericKeys()

```
public void removeAllNumericKeys()
```

Remove the key codes for the numeric keys (VK_0, VK_1, VK_2, VK_3, VK_4, VK_5, VK_6, VK_7, VK_8, VK_9). Key codes from this set which are not present in the repository will be ignored.

removeKey(int)

```
public void removeKey(int keycode)
```

The method to remove a key from the repository. Note that when an application removes a key it has to reset the listener with this new set of user events. Removing a key which is not in the repositiory has no effect.

Parameters:

`int` - the key code.

removeUserEvent(UserEvent)

```
public void removeUserEvent(UserEvent event)
```

Remove a user event from the repository.

Parameters:

`event` - the event to be removed from the repository.

Annex K (normative): DVB-J persistent storage API

Package
org.dvb.io.persistent

Class Summary	
Classes	
FileAccessPermissions	This class encapsulates file access permissions, world, Organisation and owner.
FileAttributes	This class encapsulates the attributes of a file stored in persistent storage.

org.dvb.io.persistent
FileAccessPermissions

Syntax

```
public class FileAccessPermissions

java.lang.Object
  |
  +--org.dvb.io.persistent.FileAccessPermissions
```

Description

This class encapsulates file access permissions, world, Organisation and owner. World means all applications authorised to access persistent storage. Owner means the application which created the file. Organisation is defined as applications with the same organisation id as defined elsewhere in this specification.

Constructors

FileAccessPermissions(boolean, boolean, boolean, boolean, boolean, boolean)

```
public FileAccessPermissions(boolean ReadWorldAccessRight, boolean WriteWorldAccessRight,
            boolean ReadOrganisationAccessRight, boolean WriteOrganisationAccessRight,
            boolean ReadApplicationAccessRight, boolean WriteApplicationAccessRight)
```

This contructor encodes all the file access permissions as a set of booleans.

Parameters:

`ReadWorldAccessRight` - read access for all applications

`WriteWorldAccessRight` - write access for all applications

`ReadOrganisationAccessRight` - read access for organisation

`WriteOrganisationAccessRight` - write access for organisation

`ReadApplicationAccessRight` - read access for the owner

`WriteApplicationAccessRight` - write access for the owner

Methods

---

hasReadApplicationAccessRight()

```
public boolean hasReadApplicationAccessRight()
```

Query whether this permission includes read access for the owning application

Returns:

true if the owning application can have read access, otherwise false.

---

hasReadOrganisationAccessRight()

```
public boolean hasReadOrganisationAccessRight()
```

Query whether this permission includes read access for the organisation

Returns:
true if applications in this organisation can have read access, otherwise false.

---

hasReadWorldAccessRight()

```
public boolean hasReadWorldAccessRight()
```

Query whether this permission includes read access for the world.

Returns:
true if all applications can have read access, otherwise false.

---

hasWriteApplicationAccessRight()

```
public boolean hasWriteApplicationAccessRight()
```

Query whether this permission includes write access for the owning application

Returns:
true if the owning application can have write access, otherwise false.

---

hasWriteOrganisationAccessRight()

```
public boolean hasWriteOrganisationAccessRight()
```

Query whether this permission includes write access for the organisation

Returns:
true if applications in this organisation can have read access, otherwise false.

---

hasWriteWorldAccessRight()

```
public boolean hasWriteWorldAccessRight()
```

Query whether this permission includes write access for the world.

Returns:
true if all applications can have write access, otherwise false.

---

setPermissions(boolean, boolean, boolean, boolean, boolean, boolean)

```
public void setPermissions(boolean ReadWorldAccessRight, boolean WriteWorldAccessRight,
            boolean ReadOrganisationAccessRight, boolean WriteOrganisationAccessRight,
            boolean ReadApplicationAccessRight, boolean WriteApplicationAccessRight)
```

This method allows to modify the permissions on this instance of the FileAccessPermission class.

Parameters:

`ReadWorldAccessRight` - read access for all applications

`WriteWorldAccessRight` - write access for all applications

`ReadOrganisationAccessRight` - read access for organisation

`WriteOrganisationAccessRight` - write access for organisation

`ReadApplicationAccessRight` - read access for the owner

`WriteApplicationAccessRight` - write access for the owner

org.dvb.io.persistent
FileAttributes

Syntax
```
public class FileAttributes

java.lang.Object
  |
  +--org.dvb.io.persistent.FileAttributes
```

Description
This class encapsulates the attributes of a file stored in persistent storage. The default attributes for a file are low priority, owner read / write only permissions and null expiration date.

Fields

PRIORITY_HIGH
```
public static final int PRIORITY_HIGH
```
Value for use as a file priority.

PRIORITY_LOW
```
public static final int PRIORITY_LOW
```
Value for use as a file priority.

PRIORITY_MEDIUM
```
public static final int PRIORITY_MEDIUM
```
Value for use as a file priority.

Methods

getExpirationDate()
```
public java.util.Date getExpirationDate()
```
Returns the expiration date. It will return the value used by the platform, which need not be the same as the value set.

Returns:
the expiration date

getFileAttributes(File)
```
public static FileAttributes getFileAttributes(java.io.File f)
```

Get the attributes of a file.

Parameters:
`f` - the file to use

Returns:
a copy of the attributes of a file

Throws:
`SecurityException` - if the application is denied access to the file or to directories needed to reach the file by security policy

`IOException` - if access to the file fails due to an IO error or if the file reference is not to a valid location in persistent storage

---

getPermissions()

```
public FileAccessPermissions getPermissions()
```

Returns the file access permissions

Returns:
the file access permissions

---

getPriority()

```
public int getPriority()
```

Returns the priority to use in persistent storage

Returns:
the priority

---

setExpirationDate(Date)

```
public void setExpirationDate(java.util.Date d)
```

Sets the expiration date. This field is a hint to the platform to identify the date after which a file is no longer useful as percieved by the application. The platform may choose to use a different date than the one given as a parameter.

Parameters:
`d` - the expiration date

---

setFileAttributes(FileAttributes, File)

```
public static void setFileAttributes(FileAttributes p, java.io.File f)
```

Associate a set of file attributes with a file.

Parameters:
`p` - the file attributes to use

`f` - the file to use

Throws:
`SecurityException` - if the application is either denied access to the file or directories needed to reach the file by security policy or is not authorised to modify the attributes of the file.

`IOException` - if access to the file fails due to an IO error or if the file reference is not to a valid location in persistent storage

setPermissions(FileAccessPermissions)

```
public void setPermissions(FileAccessPermissions p)
```

Sets the file access permissions.

Parameters:

`p` - the file access permissions

setPriority(int)

```
public void setPriority(int priority)
```

Sets the priority to use in persistent storage

Parameters:

`priority` - the priority to set

Annex L (normative): User Settings and Preferences API

Package
org.dvb.user

Class Summary	
Interfaces	
RetrievablePreference	The interface for preference that can be saved.
UserPreferenceChange-Listener	An application wishing to be informed of any change to a user preference implements this interface.
Classes	
Facility	A facility maps a preference's name to a single value or to an array of values.
GeneralPreference	This class defines a set of general preferences.
Preference	This abstract class defines the Preference object.
UserPreferenceChan-geEvent	This class defines the event sent to appropriate listeners when a user preference has been changed.
UserPreferencePermis-sion	This class is for user preference and setting permissions.
UserPreferences	The UserPreferences class gives access to the user preference settings.

org.dvb.user
Facility

Syntax
```
public class Facility

java.lang.Object
  |
  +--org.dvb.user.Facility
```

Description
A facility maps a preference's name to a single value or to an array of values. A facility enables an application to define the list of values supported for a specified preference. For example, if an application is available in English or French then it can create a Facility ("User Language", {"English", "French"}). When the application will retrieve the "User Language" from the general preference it will specify the associated facility in order to get a Preference which will contain a set a values compatible with those supported by the application.

Constructors

Facility(String, String)

```
public Facility(java.lang.String preference, java.lang.String value)
```

Creates a Facility with a single value. This facility can be used by an application to retrieve a preference compatible with its capabilities.

Parameters:

`preference` - a String representing the name of the preference.

`value` - a String representing the value of the preference.

Facility(String, String[])

```
public Facility(java.lang.String preference, java.lang.String[] values)
```

Creates a Facility with a set of values. This facility can be used by an application to retrieve a preference compatible with its capabilities.

Parameters:

`preference` - a String representing the name of the preference.

`values` - an array of String representing the set of values.

org.dvb.user
GeneralPreference

Syntax
```
public final class GeneralPreference extends Preference implements RetrievablePreference
```

```
java.lang.Object
  |
  +--Preference
        |
        +--org.dvb.user.GeneralPreference
```

All Implemented Interfaces:
RetrievablePreference

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 UserPreferences.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 UserPreferences.read().

Constructors

GeneralPreference(String)

```
public GeneralPreference(java.lang.String name)
```

Constructs a GeneralPreference object. A general preference maps a preference name to a list of strings.

Parameters:
`name` - the general preference name.

Throws:
`IllegalArgumentException` - if the preference's name is not supported.

org.dvb.user
Preference

Syntax
```
public abstract class Preference

java.lang.Object
  |
  +--org.dvb.user.Preference
```

Direct Known Subclasses:
GeneralPreference

Description
This abstract class defines the Preference object. A Preference maps a name to a list of favourite values. The first element in the list is the favourite value for this preference.

Constructors

Preference()

```
protected Preference()
```

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:

`a` - String object representing the name 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 position is greater than the lengh 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.

---

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.

Parameters:
`value` - a String representing the value to remove.

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.

toString()

```
public java.lang.String toString()
```

Print name and favourites.

Overrides:
java.lang.Object.toString() in class java.lang.Object

org.dvb.user
RetrievablePreference

Syntax
```
public interface RetrievablePreference
```

All Known Implementing Classes:
GeneralPreference

Description
The interface for preference that can be saved.

org.dvb.user
UserPreferenceChangeEvent

Syntax
```
public class UserPreferenceChangeEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.user.UserPreferenceChangeEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
This class defines the event sent to appropriate listeners when a user preference has been changed.

Constructors

UserPreferenceChangeEvent(String)

```
public UserPreferenceChangeEvent(java.lang.String preferenceName)
```

Constructs a new event.

Parameters:
`source` - the name of the modified preference.

Methods

---

getName()

```
public java.lang.String getName()
```

Returns the modified Preference name.

Returns:
the Preference name.

org.dvb.user
UserPreferenceChangeListener

Syntax
```
public interface UserPreferenceChangeListener
```

Description
An application wishing to be informed of any change to a user preference implements this interface.

Methods

---

receiveUserPreferenceChangeEvent(UserPreferenceChangeEvent)

```
public void receiveUserPreferenceChangeEvent(UserPreferenceChangeEvent e)
```

This method is called when a user preference changes.

Parameters:

`e` - the event notifying this event.

org.dvb.user
UserPreferencePermission

Syntax
```
public class UserPreferencePermission extends java.security.BasicPermission

java.lang.Object
  |
  +--java.security.Permission
        |
        +--java.security.BasicPermission
              |
              +--org.dvb.user.UserPreferencePermission
```

All Implemented Interfaces:
java.security.Guard, java.io.Serializable

Description
This class is for user preference and setting permissions. A UserPreferencePermission contains a name, but no actions list.

The permission name can either be "read" or "write". The "read" permission allows an application to read the user preferences and settings (using UserPreferences.read()) for which read access is not always granted. Access to the following settings/preferences is always granted: User Language, Morality Level, Font Size and Country Code.

The "write" permission allows an application to modify user preferences and settings (using UserPreferences.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.

org.dvb.user
UserPreferences

Syntax

```
public class UserPreferences

java.lang.Object
  |
  +--org.dvb.user.UserPreferences
```

Description

The UserPreferences class gives access to the user preference settings. This class provides a set of static methods that allows an application to read or save user settings. It also provides a mechanism to notify applications when a preference has been modified. The value of a user setting,retrieved with the read method, is a copy of the value that is stored in the receiver. The write method, if authorized, overwrites the stored value.

Methods

addUserPreferenceChangeListener(UserPreferenceChangeListener)

```
public static void addUserPreferenceChangeListener(UserPreferenceChangeListener l)
```

Adds a listener for changes in user preferences.

Parameters:

`l` - the listener to add.

read(RetrievablePreference)

```
public static void read(RetrievablePreference p)
```

Allows an application to read a specified user preference. This method may throw a security exception.

Parameters:

`p` - an object representing the preference to read.

read(RetrievablePreference, Facility)

```
public static void read(RetrievablePreference p, Facility facility)
```

Allows an application to read a specified user preference taking into account the facility defined by the application. If the intersection between the two sets of values is empty then the preference will have no value. This method may throw a security exception.

Parameters:

`p` - an object representing the preference to read.

removeUserPreferenceChangeListener(UserPreferenceChangeListener)

```
public static void removeUserPreferenceChangeListener(UserPreferenceChangeListener l)
```

Removes a listener for changes in user preferences. If the listener has not previously been added then this method has no effect.

Parameters:

`l` - the listener to remove.

---

write(RetrievablePreference)

```
public static void write(RetrievablePreference p)
```

Saves the specified user preference. If this method succeeds then it will change the value of this preference for all future MHP applications. This method may throw a security exception.

Parameters:

`p` - the preference to save.

Annex M (normative): SI Access API

Package org.dvb.si

Class Summary

Interfaces

DescriptorTag	This interface defines constants corresponding to the most common descriptor tags.
PMTElementaryStream	This interface represents an elementary stream of a service.
PMTService	This interface represents a particular service carried by a transport stream.
PMTStreamType	This interface defines the constants corresponding to the different stream types
SIBouquet	This interface (together with the SITransportStreamBAT interface) represents a sub-table of the Bouquet Association Table (BAT) describing a particular bouquet.
SIEvent	This interface represents a particular event within a service.
SIInformation	This interface groups the common features of SIBouquet, SINetwork, SITransportStream, SIService, PMTService, SIEvent, SITime and PMTElementaryStream.
SIIterator	Objects implementing SIIterator interface allow to browse through a set of SI objects.
SIMonitoringListener	This interface shall be implemented by using application classes in order to listen to changes in monitored SI objects.
SIMonitoringType	This interface defines the constants corresponding to the SI information type values in SIMonitoringEvent.
SINetwork	This interface (together with the SITransportStreamNIT interface) represents a sub-table of the Network Information Table (NIT) describing a particular network.
SIRetrievalListener	This interface shall be implemented by application classes in order to receive events about completion of SI requests.
SIRunningStatus	This interface defines the constants corresponding to the running status values for services and events.
SIService	This interface represents a particular service carried by a transport stream.
SIServiceType	This interface defines constants corresponding to the different service types.
SITime	This interface represents the Time and Date Table (TDT) and the (optional) Time Offset Table (TOT).
SITransportStream	This interface is the base interface for representing information about transport streams.
SITransportStreamBAT	This interface represents information about transport streams that has been retrieved from a BAT table.
SITransportStreamDescription	This interface represents the Transport Stream Description Table (TSDT).
SITransportStreamNIT	This interface represents information about transport streams that has been retrieved from a NIT table.

Classes

Class Summary

Descriptor	This class represents a descriptor within a sub-table.
SIDatabase	This class represents the root of the SI information hierarchy.
SILackOfResourcesEvent	This event is sent in response to a SI retrieval request when the resources needed for retrieving the data are not available, e.g.
SIMonitoringEvent	Objects of this class are sent to listener objects of the using application to notify that a change in the monitored information has happened.
SINotInCacheEvent	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.
SIObjectNotInTableEvent	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.
SIRequest	Object instances of this class represent SI retrieval requests made by the application.
SIRequestCancelledEvent	This event is sent in response to a SI retrieval request when the request is cancelled with the SIRequest.cancelRequest method call.
SIRetrievalEvent	This class is the base class for events about completion of a SI retrieval request.
SISuccessfulRetrieveEvent	This event is sent in response to a SI retrieval request when the retrieve request was successfully completed.
SITableNotFoundEvent	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.
SITableUpdatedEvent	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.
SIUtil	This class contains SI related utility functions.

Exceptions

SIException	This class is the root of the SI exceptions hierarchy.
SIIllegalArgumentException	This exception is thrown when one or more of the arguments passed to a method are invalid (e.g.
SIInvalidPeriodException	This exception is thrown when a specified period is invalid (for example, start time is after the end time)

org.dvb.si
Descriptor

Syntax
```
public class Descriptor

java.lang.Object
  |
  +--org.dvb.si.Descriptor
```

Description
This class represents a descriptor within a sub-table.

A descriptor consist of three fields: a tag, a contentLength and the content.

The tag uniquely identifies the descriptor type. The content length indicates the number of bytes in the content. The content consists of an array of bytes of length content length. The data represented by the content is descriptor type dependent.

See Also:
DescriptorTag

Methods

---

getByteAt(int)

```
public byte getByteAt(int index)
```

Get a particular byte within the descriptor content

Parameters:
`index` - index to the descriptor content. Value 0 corresponds to the first byte after the length field.

Returns:
The required byte

Throws:
`java.lang.IndexOutOfBoundsException` - index < 0 or index => ContentLength

---

getContent()

```
public byte[] getContent()
```

Get a copy of the content of this descriptor (everything after the length field).

Returns:
a copy of the content of the descriptor

---

getContentLength()

```
public short getContentLength()
```

This method returns the length of the descriptor content as coded in the length field of this descriptor.

Returns:

The length of the descriptor content.

---

getTag()

```
public short getTag()
```

Get the descriptor tag

Returns:
The descriptor tag (the most common values are defined in the DescriptorTag interface)

See Also:
DescriptorTag

org.dvb.si
DescriptorTag

Syntax
```
public interface DescriptorTag
```

Description
This interface defines constants corresponding to the most common descriptor tags.

See Also:
Descriptor

Fields

BOUQUET_NAME
```
public static final short BOUQUET_NAME
```

CA_IDENTIFIER
```
public static final short CA_IDENTIFIER
```

CABLE_DELIVERY_SYSTEM
```
public static final short CABLE_DELIVERY_SYSTEM
```

COMPONENT
```
public static final short COMPONENT
```

CONTENT
```
public static final short CONTENT
```

COUNTRY_AVAILABILITY
```
public static final short COUNTRY_AVAILABILITY
```

DATA_BROADCAST
```
public static final short DATA_BROADCAST
```

EXTENDED_EVENT
```
public static final short EXTENDED_EVENT
```

FREQUENCY_LIST

```
public static final short FREQUENCY_LIST
```

LINKAGE

```
public static final short LINKAGE
```

LOCAL_TIME_OFFSET

```
public static final short LOCAL_TIME_OFFSET
```

MOSAIC

```
public static final short MOSAIC
```

MULTILINGUAL_BOUQUET_NAME

```
public static final short MULTILINGUAL_BOUQUET_NAME
```

MULTILINGUAL_COMPONENT

```
public static final short MULTILINGUAL_COMPONENT
```

MULTILINGUAL_NETWORK_NAME

```
public static final short MULTILINGUAL_NETWORK_NAME
```

MULTILINGUAL_SERVICE_NAME

```
public static final short MULTILINGUAL_SERVICE_NAME
```

NETWORK_NAME

```
public static final short NETWORK_NAME
```

NVOD_REFERENCE

```
public static final short NVOD_REFERENCE
```

PARENTAL_RATING

```
public static final short PARENTAL_RATING
```

PARTIAL_TRANSPORT_STREAM

```
public static final short PARTIAL_TRANSPORT_STREAM
```

PRIVATE_DATA_SPECIFIER

```
public static final short PRIVATE_DATA_SPECIFIER
```

SATELLITE_DELIVERY_SYSTEM

```
public static final short SATELLITE_DELIVERY_SYSTEM
```

SERVICE

```
public static final short SERVICE
```

SERVICE_LIST

```
public static final short SERVICE_LIST
```

SERVICE_MOVE

```
public static final short SERVICE_MOVE
```

SHORT_EVENT

```
public static final short SHORT_EVENT
```

SHORT_SMOOTHING_BUFFER

```
public static final short SHORT_SMOOTHING_BUFFER
```

STREAM_IDENTIFIER

```
public static final short STREAM_IDENTIFIER
```

STUFFING

```
public static final short STUFFING
```

SUBTITLING

```
public static final short SUBTITLING
```

TELEPHONE

```
public static final short TELEPHONE
```

TELETEXT

```
public static final short TELETEXT
```

TERRESTRIAL_DELIVERY_SYSTEM

```
public static final short TERRESTRIAL_DELIVERY_SYSTEM
```

TIME_SHIFTED_EVENT

```
public static final short TIME_SHIFTED_EVENT
```

TIME_SHIFTED_SERVICE

```
public static final short TIME_SHIFTED_SERVICE
```

org.dvb.si
PMTElementaryStream

Syntax
```
public interface PMTElementaryStream extends SIInformation
```

All Superinterfaces:
SIInformation

Description
This interface represents an elementary stream of a service.

For each running service there is a PMT describing the elementary streams of the service. An object that implements this interface represents one such elementary stream. Each object that implements the PMTElementaryStream interface is identified by the combination of the identifiers original_network_id, transport_stream_id, service_id, component_tag (or elementary_PID).

See Also:
PMTService, PMTStreamType

Methods

---

getComponentTag()

```
public int getComponentTag()
```

Get the component tag identifier.

Returns:
The component tag. If the elementary stream does not have an associated component tag, this method returns -2.

---

getDvbLocator()

```
public org.davic.net.dvb.DvbLocator getDvbLocator()
```

Gets a DvbLocator that identifies this elementary stream

Returns:
The DvbLocator of this elementary stream

---

getElementaryPID()

```
public short getElementaryPID()
```

Get the elementary PID.

Returns:
The PID the data of elementary stream is sent on in the transport stream.

getOriginalNetworkID()

```
public int getOriginalNetworkID()
```

Get the original network identification identifier.

Returns:
The original network identification.

getServiceID()

```
public int getServiceID()
```

Get the service identification identifier.

Returns:
The service identification.

getStreamType()

```
public byte getStreamType()
```

Get the stream type of this elemetary stream.

Returns:
The stream type (some of the possible values are defined in the PMTStreamType interface).

See Also:
PMTStreamType

getTransportStreamID()

```
public int getTransportStreamID()
```

Get the transport stream identification identifier.

Returns:
The transport stream identification.

org.dvb.si
PMTService

Syntax
```
public interface PMTService extends SIInformation
```

All Superinterfaces:
SIInformation

Description
This interface represents a particular service carried by a transport stream. The information is retrieved from the PMT table.

Each object that implements the PMTService interface is identified by the combination of the following identifiers: original_network_id, transport_stream_id, service_id.

Methods

---

getDvbLocator()

```
public org.davic.net.dvb.DvbLocator getDvbLocator()
```

Gets a DvbLocator that identifies this service

Returns:
The DvbLocator of this service

---

getOriginalNetworkID()

```
public int getOriginalNetworkID()
```

Get the original network identification.

Returns:
The original network identification identifier.

---

getPcrPid()

```
public int getPcrPid()
```

Get the PCR pid.

Returns:
The PCR pid.

---

getServiceID()

```
public int getServiceID()
```

Get the service identification.

Returns:
The service identification identifier.

getTransportStreamID()

```
public int getTransportStreamID()
```

Get the transport stream identification.

Returns:
The transport stream identification identifier.

retrievePMTElementaryStreams(short, Object, SIRetrievalListener, short[])

```
public SIRequest retrievePMTElementaryStreams(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, short[] somePMTDescriptorTags)
```

Retrieve information associated with the elementary streams which compose this service from the Program Map Table (PMT).

The SIIterator that is returned with the event when the request completes successfully will contain one or more objects that implement the PMTElementaryStream interface.

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 non applicable tag values are ignored.

Returns:
An SIRequest object

Throws:
SIIllegalArgumentException - thrown if the retrieveMode is invalid

See Also:
SIRequest, SIRetrievalListener, PMTElementaryStream

org.dvb.si
PMTStreamType

Syntax
```
public interface PMTStreamType
```

Description
This interface defines the constants corresponding to the different stream types

See Also:
PMTElementaryStream, getStreamType()

Fields

MPEG1_AUDIO
```
public static final byte MPEG1_AUDIO
```

MPEG1_VIDEO
```
public static final byte MPEG1_VIDEO
```

MPEG2_AUDIO
```
public static final byte MPEG2_AUDIO
```

MPEG2_VIDEO
```
public static final byte MPEG2_VIDEO
```

org.dvb.si
SIBouquet

Syntax
```
public interface SIBouquet extends SIInformation
```

All Superinterfaces:
SIInformation

Description
This interface (together with the SITransportStreamBAT interface) represents a sub-table of the Bouquet Association Table (BAT) describing a particular bouquet.

Each object that implements the SIBouquet interface is identified by the identifier bouquet_id.

See Also:
SITransportStreamBAT

Methods

---

getBouquetID()

```
public int getBouquetID()
```

Get the identification.

Returns:
The bouquet identification of this bouquet.

---

getDescriptorTags()

```
public short[] getDescriptorTags()
```

This method defines extra semantics for the SIInformation.getDescriptorTags method. If the BAT sub-table on which this SIBouquet object is based consists of multiple sections, then this method returns the descriptor tags in the order they appear when concatenating the descriptor loops of the different sections.

Overrides:
getDescriptorTags() in interface SIInformation

See Also:
SIInformation, getDescriptorTags()

---

getName()

```
public java.lang.String getName()
```

This method returns the name of this bouquet. The name is extracted from the bouquet_name_descriptor or optionally from the multilingual_bouquet_name_descriptor. When this information is not available "" is returned. All control characters as defined in ETR 211 are ignored.

For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation

Returns:
The bouquet name of this bouquet.

---

getShortBouquetName()

```
public java.lang.String getShortBouquetName()
```

This method returns the short name (ETR 211) of this bouquet without emphasis marks. The name is extracted from the bouquet_name_descriptor or optionally from the multilingual_bouquet_name_descriptor. When this information is not available "" is returned. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

Returns:
The short bouquet name of this bouquet.

---

getSIServiceLocators()

```
public org.davic.net.dvb.DvbLocator[] getSIServiceLocators()
```

Get a list of DvbLocators identifying the services that belong to the bouquet.

Returns:
An array of DvbLocators identifying the services

See Also:
`org.davic.net.dvb.DvbLocator`, SIService

---

retrieveDescriptors(short, Object, SIRetrievalListener)

```
public SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener)
```

This method defines extra semantics for the SIInformation.retrieveDescriptors method (first prototype). If the BAT sub-table on which this SIBouquet object is based consists of multiple sections, then this method returns the requested descriptors in the order they appear when concatenating the descriptor loops of the different sections.

Overrides:
retrieveDescriptors(short, Object, SIRetrievalListener) in interface SIInformation

Throws:
SIIllegalArgumentException

See Also:
SIInformation, retrieveDescriptors(short, Object, SIRetrievalListener)

---

retrieveDescriptors(short, Object, SIRetrievalListener, short[])

```
public SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, short[] someDescriptorTags)
```

This method defines extra semantics for the SIInformation.retrieveDescriptors method (second prototype). If the BAT sub-table on which this 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 SIInformation

Throws:

SIIllegalArgumentException

See Also:

SIInformation, retrieveDescriptors(short, Object, SIRetrievalListener, short[])

---

retrieveSIBouquetTransportStreams(short, Object, SIRetrievalListener, short[])

```
public SIRequest retrieveSIBouquetTransportStreams(short retrieveMode,
            java.lang.Object appData, SIRetrievalListener listener,
            short[] someDescriptorTags)
```

Retrieve information associated with transport streams belonging to the bouquet.

The SIIterator that is returned with the event when the request completes successfully will contain one or more objects that implement the SITransportStreamBAT interface.

Parameters:

`retrievemode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, The application is interested in all descriptors. If someDescriptorTags is null, the application is not interested in descriptors. All non applicable tag values are ignored.

Returns:

An SIRequest object

Throws:

SIIllegalArgumentException - thrown if the retrieveMode is invalid

See Also:

SIRequest, SIRetrievalListener, SITransportStreamBAT, DescriptorTag

org.dvb.si
SIDatabase

Syntax
```
public class SIDatabase

java.lang.Object
  |
  +--org.dvb.si.SIDatabase
```

Description
This class represents the root of the SI information hierarchy. There is one SIDatabase per network interface. In a system with a single network interface there is only one SIDatabase object.

Methods

---

addBouquetMonitoringListener(SIMonitoringListener, int)

```
public void addBouquetMonitoringListener(SIMonitoringListener listener, int bouquetId)
```

Initiate monitoring of the bouquet information. When the bouquet information changes, an event will be delivered to the registered listener object.

How the monitoring is performed is implementation dependent and especially does not necessarily need to be continuous. The event will be delivered as soon as the implementation notices the change which might have some delay relative to when the change was actually made in the stream due to resources for the monitoring being scheduled between the monitoring activities of different tables.

The monitoring stops silently when the network interface with which this SIDatabase object is associated starts tuning to another transport stream.

Parameters:

`listener` - listener object that will receive events when a change in the information is detected.

`bouquetId` - bouquet identifier of the bouquet whose information will be monitored.

Throws:

SIIllegalArgumentException - thrown if the identifiers are invalid (e.g. out of range)

See Also:

SIMonitoringListener, SIMonitoringEvent

---

addEventPresentFollowingMonitoringListener(SIMonitoringListener, int, int, int)

```
public void addEventPresentFollowingMonitoringListener(SIMonitoringListener listener,
            int originalNetworkId, int transportStreamId, int serviceId)
```

Initiate monitoring of information in the EIT related to present and following events. When the information related to those events changes, an event will be delivered to the registered listener object.

The scope of the monitoring is determined by the original network identifier, transport stream identifier and service identifier. The listener will be notified about the change of the information in any present and following event within that scope.

How the monitoring is performed is implementation dependent and especially does not necessarily need to be continuous. The event will be delivered as soon as the implementation notices the change which might have some delay relative to when the change was actually made in the stream due to resources for the monitoring being scheduled between the monitoring activities of different tables.

The monitoring stops silently when the network interface with which this SIDatabase object is associated starts tuning to another transport stream.

Parameters:

`listener` - listener object that will receive events when a change in the information is detected.

`originalNetworkId` - original network identifier specifying the scope of the monitoring.

`transportStreamId` - transport stream identifier specifying the scope of the monitoring.

`serviceId` - service identifier specifying the scope of the monitoring

Throws:

SIIllegalArgumentException - thrown if the identifiers are invalid (e.g. out of range)

See Also:

SIMonitoringListener, SIMonitoringEvent

---

addEventScheduleMonitoringListener(SIMonitoringListener, int, int, int, Date, Date)

```
public void addEventScheduleMonitoringListener(SIMonitoringListener listener,
            int originalNetworkId, int transportStreamId, int serviceId,
            java.util.Date startTime, java.util.Date endTime)
```

Initiate monitoring of information in the EIT related to scheduled events. When the information related to those events changes, an event will be delivered to the registered listener object.

The scope of the monitoring is determined by the original network identifier, transport stream identifier, service identifier, start time and end time of the schedule period. The listener will be notified about the change of the information in any scheduled event within that scope.

How the monitoring is performed is implementation dependent and especially does not necessarily need to be continuous. The event will be delivered as soon as the implementation notices the change which might have some delay relative to when the change was actually made in the stream due to resources for the monitoring being scheduled between the monitoring activities of different tables.

The monitoring stops silently when the network interface with which this SIDatabase object is associated starts tuning to another transport stream.

Parameters:

`listener` - listener object that will receive events when a change in the information is detected.

`originalNetworkId` - original network identifier specifying the scope of the monitoring.

`transportStreamId` - transport stream identifier specifying the scope of the monitoring.

`serviceId` - service identifier specifying the scope of the monitoring

`startTime` - start time of the schedule period

`endTime` - end time of the schedule period

Throws:

SIIllegalArgumentException - thrown if the identifiers are invalid (e.g. out of range)

SIInvalidPeriodException - thrown if the period is invalid (e.g. end time before start time)

See Also:

SIMonitoringListener, SIMonitoringEvent

addNetworkMonitoringListener(SIMonitoringListener, int)

```
public void addNetworkMonitoringListener(SIMonitoringListener listener, int networkId)
```

Initiate monitoring of the network information. When the network information changes, an event will be delivered to the registered listener object.

How the monitoring is performed is implementation dependent and especially does not necessarily need to be continuous. The event will be delivered as soon as the implementation notices the change which might have some delay relative to when the change was actually made in the stream due to resources for the monitoring being scheduled between the monitoring activities of different tables.

The monitoring stops silently when the network interface with which this SIDatabase object is associated starts tuning to another transport stream.

Parameters:

`listener` - listener object that will receive events when a change in the information is detected.

`networkId` - network identifier of the network whose information will be monitored.

Throws:

SIIllegalArgumentException - thrown if the identifiers are invalid (e.g. out of range)

See Also:

SIMonitoringListener, SIMonitoringEvent

addPMTServiceMonitoringListener(SIMonitoringListener, int, int, int)

```
public void addPMTServiceMonitoringListener(SIMonitoringListener listener,
            int originalNetworkId, int transportStreamId, int serviceId)
```

Initiate monitoring of information in the PMT related to a service. When the information related to a service changes, an event will be delivered to the registered listener object.

How the monitoring is performed is implementation dependent and especially does not necessarily need to be continuous. The event will be delivered as soon as the implementation notices the change which might have some delay relative to when the change was actually made in the stream due to resources for the monitoring being scheduled between the monitoring activities of different tables.

The monitoring stops silently when the network interface with which this SIDatabase object is associated starts tuning to another transport stream.

Parameters:

`listener` - listener object that will receive events when a change in the information is detected.

`originalNetworkId` - original network identifier of the service

`transportStreamId` - transport stream identifier of the service

`serviceId` - service identifier specifying the service whose information will be monitored

Throws:

SIIllegalArgumentException - thrown if the identifiers are invalid (e.g. out of range)

See Also:

SIMonitoringListener, SIMonitoringEvent

addServiceMonitoringListener(SIMonitoringListener, int, int)

```
public void addServiceMonitoringListener(SIMonitoringListener listener,
            int originalNetworkId, int transportStreamId)
```

Initiate monitoring of information in the SDT related to services. When the information related to services changes, an event will be delivered to the registered listener object.

The scope of the monitoring is determined by the original network identifier and transport stream identifier. The listener will be notified about the change of the information in any service within that scope.

How the monitoring is performed is implementation dependent and especially does not necessarily need to be continuous. The event will be delivered as soon as the implementation notices the change which might have some delay relative to when the change was actually made in the stream due to resources for the monitoring being scheduled between the monitoring activities of different tables.

The monitoring stops silently when the network interface with which this SIDatabase object is associated starts tuning to another transport stream.

Parameters:

`listener` - listener object that will receive events when a change in the information is detected.

`originalNetworkId` - original network identifier specifying the scope of the monitoring.

`transportStreamId` - transport stream identifier specifying the scope of the monitoring.

Throws:

SIIllegalArgumentException - thrown if the identifiers are invalid (e.g. out of range)

See Also:

SIMonitoringListener, SIMonitoringEvent

---

getSIDatabase()

```
public static SIDatabase[] getSIDatabase()
```

Return an array of SIDatabase objects (one object per network interface). In a system with one network interface, the length of this array will be one. The network interface of each SIDatabase is used as data source for all new data accessed by this SIDatabase or SIInformation instances obtained from it.

This is the first method to be called to access the DVB-SI API. The returned SIDatabase objects provide the access point to the DVB-SI information.

Returns:

An array of SIDatabase objects, one per network interface.

---

removeBouquetMonitoringListener(SIMonitoringListener, int)

```
public void removeBouquetMonitoringListener(SIMonitoringListener listener, int bouquetId)
```

Removes the registration of an event listener for bouquet information monitoring.

Parameters:

`listener` - listener object that has previously been registered

`networkId` - bouquet identifier of the bouquet whose information has been requested to be monitored

Throws:

SIIllegalArgumentException - thrown if the identifiers are invalid (e.g. out of range)

See Also:

SIMonitoringListener, SIMonitoringEvent

removeEventPresentFollowingMonitoringListener(SIMonitoringListener, int, int, int)

```
public void removeEventPresentFollowingMonitoringListener(SIMonitoringListener listener,
            int originalNetworkId, int transportStreamId, int serviceId)
```

Removes the registration of an event listener for monitoring information related to present and following events

Parameters:

`listener` - listener object that has previously been registered

`originalNetworkId` - original network identifier specifying the scope of the monitoring.

`transportStreamId` - transport stream identifier specifying the scope of the monitoring.

`serviceId` - service identifier specifying the scope of the monitoring

Throws:

SIIllegalArgumentException - thrown if the identifiers are invalid (e.g. out of range)

See Also:

SIMonitoringListener, SIMonitoringEvent

removeEventScheduleMonitoringListener(SIMonitoringListener, int, int, int)

```
public void removeEventScheduleMonitoringListener(SIMonitoringListener listener,
            int originalNetworkId, int transportStreamId, int serviceId)
```

Removes the registration of an event listener for monitoring information related to scheduled events for all periods

Parameters:

`listener` - listener object that has previously been registered

`originalNetworkId` - original network identifier specifying the scope of the monitoring.

`transportStreamId` - transport stream identifier specifying the scope of the monitoring.

`serviceId` - service identifier specifying the scope of the monitoring

Throws:

SIIllegalArgumentException - thrown if the identifiers are invalid (e.g. out of range)

See Also:

SIMonitoringListener, SIMonitoringEvent

removeNetworkMonitoringListener(SIMonitoringListener, int)

```
public void removeNetworkMonitoringListener(SIMonitoringListener listener, int networkId)
```

Removes the registration of an event listener for network information monitoring.

Parameters:

`listener` - listener object that has previously been registered

`networkId` - network identifier of the network 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

removePMTServiceMonitoringListener(SIMonitoringListener, int, int, int)

```
public void removePMTServiceMonitoringListener(SIMonitoringListener listener,
            int originalNetworkId, int transportStreamId, int serviceId)
```

Removes the registration of an event listener for monitoring information in the PMT related to a service.

Parameters:

`listener` - listener object that has previously been registered

`originalNetworkId` - original network identifier of the service

`transportStreamId` - transport stream identifier of the service

`serviceId` - service identifier specifying the service whose information has been requested to be monitored

Throws:

SIIllegalArgumentException - thrown if the identifiers are invalid (e.g. out of range)

See Also:

SIMonitoringListener, SIMonitoringEvent

removeServiceMonitoringListener(SIMonitoringListener, int, int)

```
public void removeServiceMonitoringListener(SIMonitoringListener listener,
            int originalNetworkId, int transportStreamId)
```

Removes the registration of an event listener for monitoring information related to services.

Parameters:

`listener` - listener object that has previously been registered

`originalNetworkId` - original network identifier specifying the scope of the monitoring.

`transportStreamId` - transport stream identifier specifying the scope of the monitoring.

Throws:

SIIllegalArgumentException - thrown if the identifiers are invalid (e.g. out of range)

See Also:

SIMonitoringListener, SIMonitoringEvent

retrieveActualSINetwork(short, Object, SIRetrievalListener, short[])

```
public SIRequest retrieveActualSINetwork(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, short[] someDescriptorTags)
```

Retrieve information associated with the actual network. The actual network is the network carrying the transport stream currently selected by the network interface connected to this SIDatabase.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SINetwork interface.

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 non applicable tag values are ignored

Returns:
An SIRequest object

Throws:
SIIllegalArgumentException - thrown if the retrieveMode is invalid

See Also:
SIRequest, SIRetrievalListener, SINetwork, DescriptorTag

---

retrieveActualSIServices(short, Object, SIRetrievalListener, short[])

```
public SIRequest retrieveActualSIServices(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, short[] someDescriptorTags)
```

Retrieve information associated with the actual services. The actual services are the services in the transport stream currently selected by the network interface connected to this SIDatabase.

The SIIterator that is returned with the event when the request completes successfully will contain 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 non applicable tag values are ignored.

Returns:
An SIRequest object

Throws:
SIIllegalArgumentException - thrown if the retrieveMode is invalid

See Also:
SIRequest, SIRetrievalListener, SIService, DescriptorTag

---

retrieveActualSITransportStream(short, Object, SIRetrievalListener, short[])

```
public SIRequest retrieveActualSITransportStream(short retrieveMode,
            java.lang.Object appData, SIRetrievalListener listener,
            short[] someDescriptorTags)
```

Retrieve information associated with the actual transport stream. The actual transport stream is the transport stream currently selected by the network interface connected to this SIDatabase.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SITransportStreamNIT interface.

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 non applicable tag values are ignored.

Returns:

An SIRequest object

Throws:

SIIllegalArgumentException - thrown if the retrieveMode is invalid

See Also:

SIRequest, SIRetrievalListener, SITransportStream, DescriptorTag

---

retrievePMTElementaryStreams(short, Object, SIRetrievalListener, DvbLocator, short[])

```
public SIRequest retrievePMTElementaryStreams(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, org.davic.net.dvb.DvbLocator dvbLocator,
            short[] someDescriptorTags)
```

Retrieve PMT elementary stream information associated with components of a service. The required component(s) can be specified by its DVB locator.

The SIIterator that is returned with the event when the request completes successfully will contain objects that implement the PMTElementaryStream 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.

`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 non applicable tag values are ignored.

Returns:

An SIRequest object

Throws:

SIIllegalArgumentException - thrown if the retrieveMode is invalid or if the locator is invalid and does not identify one or more service components

See Also:
SIRequest, SIRetrievalListener, SIService, DescriptorTag

---

retrievePMTElementaryStreams(short, Object, SIRetrievalListener, int, int, short[])

```
public SIRequest retrievePMTElementaryStreams(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, int serviceId, int componentTag,
            short[] someDescriptorTags)
```

Retrieve PMT elementary stream information associated with components of a service from the actual transport stream of this SIDatabase object. The elementary streams can be specified by their identification (-1 means 'any' for componentTag)

The SIIterator that is returned with the event when the request completes successfully will contain objects that implement the PMTElementaryStream 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.

`serviceId` - Identification of the elementary streams to be retrieved: service identifier

`componentTag` - Identification of the elementary streams to be retrieved: component tag (-1 as wilcard allowed)

`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 non applicable tag values are ignored.

Returns:
An SIRequest object

Throws:
SIIllegalArgumentException - thrown if the retrieveMode is invalid or the numeric identifiers are out of range

See Also:
SIRequest, SIRetrievalListener, SIService, DescriptorTag

---

retrievePMTService(short, Object, SIRetrievalListener, DvbLocator, short[])

```
public SIRequest retrievePMTService(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, org.davic.net.dvb.DvbLocator dvbLocator,
            short[] someDescriptorTags)
```

Retrieve PMT information associated with a service. The required service can be specified by its DVB locator.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the PMTService interface.

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 non applicable tag values are ignored.

Returns:

An SIRequest object

Throws:

SIIllegalArgumentException - thrown if the retrieveMode is invalid or the locator is invalid and does not identify a service

See Also:

SIRequest, SIRetrievalListener, SIService, DescriptorTag

---

retrievePMTServices(short, Object, SIRetrievalListener, int, short[])

```
public SIRequest retrievePMTServices(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, int serviceId, short[] someDescriptorTags)
```

Retrieve PMT information associated with services from the actual transport stream of this SIDatabase object. The required services can be specified by their identification (-1 means 'any' for serviceId)

The SIIterator that is returned with the event when the request completes successfully will contain objects that implement the PMTService 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.

`serviceId` - Identification of the services to be retrieved: service identifier (-1 as wilcard allowed)

`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 non applicable tag values are ignored.

Returns:

An SIRequest object

Throws:

SIIllegalArgumentException - thrown if the retrieveMode is invalid or the numeric identifiers are out of range

See Also:

SIRequest, SIRetrievalListener, SIService, DescriptorTag

---

retrieveSIBouquets(short, Object, SIRetrievalListener, int, short[])

```
public SIRequest retrieveSIBouquets(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, int bouquetId, short[] someDescriptorTags)
```

Retrieve information associated with bouquets. A bouquet can be specified by its identification (when bouquetId is set to -1, all bouquets are retrieved)

The SIIterator that is returned with the event when the request completes successfully will contain objects that implement the SIBouquet 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.

`bouquetId` - Identifier of the bouquet to be retrieved.

`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 non applicable tag values are ignored

Returns:

An SIRequest object

Throws:

SIIllegalArgumentException - thrown if the retrieveMode is invalid or the numeric identifiers are out of range

See Also:

SIRequest, SIRetrievalListener, SIBouquet, DescriptorTag

---

retrieveSINetworks(short, Object, SIRetrievalListener, int, short[])

```
public SIRequest retrieveSINetworks(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, int networkId, short[] someDescriptorTags)
```

Retrieve information associated with networks. A network can be specified by its identification (when networkId is set to -1, all networks are retrieved)

The SIIterator that is returned with the event when the request completes successfully will contain objects that implement the SINetwork 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.

`networkId` - Identification of the network to be retrieved (-1 as wildcard allowed)

`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 non applicable tag values are ignored.

Returns:
An SIRequest object

Throws:
SIIllegalArgumentException - thrown if the retrieveMode is invalid or the numeric identifiers are out of range

See Also:
SIRequest, SIRetrievalListener, SINetwork, DescriptorTag

---

retrieveSIService(short, Object, SIRetrievalListener, DvbLocator, short[])

```
public SIRequest retrieveSIService(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, org.davic.net.dvb.DvbLocator dvbLocator,
            short[] someDescriptorTags)
```

Retrieve information associated with a service. The required service can be specified by its DVB locator.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SIService interface.

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 non applicable tag values are ignored.

Returns:
An SIRequest object

Throws:
SIIllegalArgumentException - thrown if the retrieveMode is invalid or the locator is invalid and does not identify a service

See Also:

SIRequest, SIRetrievalListener, SIService, DescriptorTag

---

retrieveSIServices(short, Object, SIRetrievalListener, int, int, int, short[])

```
public SIRequest retrieveSIServices(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, int originalNetworkId, int transportStreamId,
            int serviceId, short[] someDescriptorTags)
```

Retrieve information associated with services. The required services can be specified by their identification (-1 means 'any' for tranportStreamId and serviceId)

The SIIterator that is returned with the event when the request completes successfully will contain objects that implement the SIService interface.

Parameters:

`retrievemode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`originalNetworkId` - Identification of the services to be retrieved: original network identifier

`transportStreamId` - Identification of the services to be retrieved: transport stream identifier (-1 as wilcard allowed)

`serviceId` - Identification of the services to be retrieved: service identifier (-1 as wilcard allowed)

`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 non applicable tag values are ignored.

Returns:

An SIRequest object

Throws:

SIIllegalArgumentException - thrown if the retrieveMode is invalid or the numeric identifiers are out of range

See Also:

SIRequest, SIRetrievalListener, SIService, DescriptorTag

---

retrieveSITimeFromTDT(short, Object, SIRetrievalListener)

```
public SIRequest retrieveSITimeFromTDT(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener)
```

Retrieve information associated with time from the Time and Date Table (TDT) from the actual transport stream.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SITime interface.

Parameters:

`retrievemode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

Returns:
An SIRequest object

Throws:
SIIllegalArgumentException - thrown if the retrieveMode is invalid

See Also:
SIRequest, SIRetrievalListener, SITime

---

retrieveSITimeFromTOT(short, Object, SIRetrievalListener, short[])

```
public SIRequest retrieveSITimeFromTOT(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, short[] someDescriptorTags)
```

Retrieve information associated with time from the Time Offset Table (TOT) from the actual transport stream. The time information will be accompanied with offset information

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SITime interface.

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 non applicable tag values are ignored.

Returns:
An SIRequest object

Throws:
SIIllegalArgumentException - thrown if the retrieveMode is invalid

See Also:
SIRequest, SIRetrievalListener, SITime

---

retrieveSITransportStreamDescription(short, Object, SIRetrievalListener, short[])

```
public SIRequest retrieveSITransportStreamDescription(short retrieveMode,
            java.lang.Object appData, SIRetrievalListener listener,
            short[] someDescriptorTags)
```

Retrieve the SITransportStreamDescription object representing the information of the TSDT table in the actual transport stream of this SIDatabase object.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SITransportStreamDescription interface.

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 non applicable tag values are ignored.

Returns:

An SIRequest object

Throws:

SIIllegalArgumentException - thrown if the retrieveMode is invalid

See Also:

SIRequest, SIRetrievalListener, SITransportStreamDescription, DescriptorTag

org.dvb.si
SIEvent

Syntax
```
public interface SIEvent extends SIInformation
```

All Superinterfaces:
SIInformation

Description
This interface represents a particular event within a service.

Each object that implements the SIEvent interface is defined by the combination of the identifiers original_network_id, transport_stream_id, service_id, event_id.

See Also:
SIService

Methods

---

getContentNibbles()

```
public byte[] getContentNibbles()
```

This method returns the content nibbles related to the event. This information is extracted from the content_descriptor. If this descriptor is not present an empty array is returned (array with length 0). The return value is an array, each array element describes one content nibble. In each nibble the level 1 content nibbles occupy the four most significant bits of the returned bytes, level 2 content nibbles the four least significant bits.

Returns:
The content nibbles related to the event; level 1 content nibbles occupy the four most significant bits of the returned bytes, level 2 content nibbles the four least significant bits.

---

getDuration()

```
public long getDuration()
```

Get the duration of this event.

Returns:
The duration in milliseconds.

---

getDvbLocator()

```
public org.davic.net.dvb.DvbLocator getDvbLocator()
```

Gets a DvbLocator that identifies this event.

Returns:
The DvbLocator of this event

getEventID()

```
public int getEventID()
```

Get the event identification.

Returns:
The event identification.

getFreeCAMode()

```
public boolean getFreeCAMode()
```

Get the free_CA_mode value for this event, false indicates none of the component streams of this event are scrambled.

Returns:
The free_CA_mode value.

getLevel1ContentNibbles()

```
public byte[] getLevel1ContentNibbles()
```

This method returns the level 1 content nibbles of this event. This information is extracted from the content_descriptor. If this descriptor is not present an empty array is returned (array with length 0). The return value is an array, each array element describes one content nibble. In each nibble the data occupies the four least significant bits of the returned bytes.

Returns:
All level 1 content nibbles related to the event.

getName()

```
public java.lang.String getName()
```

This method returns the name of this event. The name is extracted from a short_event_descriptor. When this information is not available "" is returned. All control characters as defined in ETR 211 are ignored. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

Returns:
The event name of this event.

getOriginalNetworkID()

```
public int getOriginalNetworkID()
```

Get the original network identification identifier.

Returns:
The original network identification.

getRunningStatus()

```
public byte getRunningStatus()
```

Get the running status of this event.

Returns:
The running status (the possible values are defined in the SIRunningStatus interface).

See Also:
SIRunningStatus

---

getServiceID()

```
public int getServiceID()
```

Get the service identification identifier.

Returns:
The service identification.

---

getShortDescription()

```
public java.lang.String getShortDescription()
```

This method returns the description of this event. The description is extracted from a short_event_descriptor. When this information is not available, "" is returned. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation

Returns:
The short description of this event.

---

getShortEventName()

```
public java.lang.String getShortEventName()
```

This method returns the short event name (ETR 211) of this event without emphasis marks. The name is extracted from a short_event_descriptor. When this information is not available "" is returned. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

Returns:
The short event name of this event.

---

getStartTime()

```
public java.util.Date getStartTime()
```

Get the start time of this event in UTC time.

Returns:
The start time of this event.

---

getTransportStreamID()

```
public int getTransportStreamID()
```

Get the transport stream identification identifier.

Returns:
The transport stream identification.

---

retrieveSIService(short, Object, SIRetrievalListener, short[])

```
public SIRequest retrieveSIService(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, short[] someDescriptorTags)
```

This method retrieves the SIService object representing the service the event, represented by this SIEvent, is part of.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SIService interface.

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 non applicable tag values are ignored

Returns:

An SIRequest object

Throws:

SIIllegalArgumentException - thrown if the retrieveMode is invalid

See Also:

SIRequest, SIRetrievalListener, SIService, DescriptorTag

org.dvb.si
SIException

Syntax
```
public abstract class SIException extends java.lang.Exception

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.dvb.si.SIException
```

Direct Known Subclasses:
SIIllegalArgumentException, SIInvalidPeriodException

All Implemented Interfaces:
java.io.Serializable

Description
This class is the root of the SI exceptions hierarchy.

Constructors

SIException()

```
public SIException()
```

SIException(String)

```
public SIException(java.lang.String reason)
```

Constructor for the SI exception with a specified reason

Parameters:

`reason` - the reason why the exception was raised

org.dvb.si
SIIllegalArgumentException

Syntax
```
public class SIIllegalArgumentException extends SIException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--SIException
                    |
                    +--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()
```

SIIllegalArgumentException(String)

```
public SIIllegalArgumentException(java.lang.String reason)
```

Constructor for the exception with a specified reason

Parameters:
`reason` - the reason why the exception was raised

org.dvb.si
SIInformation

Syntax
```
public interface SIInformation
```

All Known Subinterfaces:
PMTElementaryStream, PMTService, SIBouquet, SIEvent, SINetwork, SIService, SITime, SITransportStream, SITransportStreamBAT, SITransportStreamDescription, SITransportStreamNIT

Description
This interface groups the common features of SIBouquet, SINetwork, SITransportStream, SIService, PMTService, SIEvent, SITime and PMTElementaryStream.

Each SIInformation instance represents a sub-table (part). Any method accessing descriptors will retrieve descriptors from the same sub-table version as the SIInformation instance. When this version is no longer available, an SITableUpdatedEvent is returned.

See Also:
SIBouquet, SINetwork, SITransportStream, SIService, PMTService, SIEvent, SITime, PMTElementaryStream

Fields

FROM_CACHE_ONLY

```
public static final short FROM_CACHE_ONLY
```

Constant for retrieve mode parameter of the retrieve methods. When FROM_CACHE_ONLY mode is specified, the data will be retrieved only if it is in the cache. Otherwise, SINotInCacheEvent will be delivered to the listener. No stream access is done in this case.

FROM_CACHE_OR_STREAM

```
public static final short FROM_CACHE_OR_STREAM
```

Constant for retrieve mode parameter of the retrieve methods. When FROM_CACHE_OR_STREAM mode is specified, the data will be retrieved from cache if it is present in the cache, otherwise it will be retrieved from the stream.

FROM_STREAM_ONLY

```
public static final short FROM_STREAM_ONLY
```

Constant for retrieve mode parameter of the retrieve methods. When FROM_STREAM_ONLY mode is specified, the data will be retrieved directly from the stream and no cache access is tried first. This mode is meaningful only if the application knows that the information is not in the cache or that the information in the cache is no longer valid, but the implementation of the SI database may not be aware of the invalidity of the cached data. If the application has got the notification of the existence of an updated version through the listener mechanism in this API, the implementation of the SI data-

base 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' distiction, 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.

Returns:

The tags of the descriptors actually broadcast for the object (identified by their tags).

See Also:

DescriptorTag

getSIDatabase()

```
public SIDatabase getSIDatabase()
```

Return the root of the hierarchy the object that implements this interface belongs to.

Returns:

The root of the hierarchy.

getUpdateTime()

```
public java.util.Date getUpdateTime()
```

Return the time when the information contained in the object that implements this interface was last updated.

Returns:
The date of the last update.

---

retrieveDescriptors(short, Object, SIRetrievalListener)

```
public SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener)
```

This method retrieves all descriptors in the order the descriptors are broadcast.

The SIIterator that is returned with the event when the request completes successfully will contain Descriptor objects.

Parameters:
`retrievemode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

Returns:
An SIRequest object

Throws:
SIIllegalArgumentException - thrown if the retrieveMode is invalid

See Also:
SIRequest, SIRetrievalListener, Descriptor, SIIterator

---

retrieveDescriptors(short, Object, SIRetrievalListener, short[])

```
public SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, short[] someDescriptorTags)
```

Retrieve a set of descriptors. This method retrieves all or a set of descriptors in the order the descriptors are broadcast.

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

The SIIterator that is returned with the event when the request completes successfully will contain Descriptor 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` - 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 non applicable tag values are ignored.

Returns:
An SIRequest object

Throws:
SIIllegalArgumentException - thrown if the retrieveMode is invalid

See Also:
SIRequest, SIRetrievalListener, Descriptor, SIIterator, DescriptorTag

org.dvb.si
SIInvalidPeriodException

Syntax
```
public class SIInvalidPeriodException extends SIException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--SIException
                    |
                    +--org.dvb.si.SIInvalidPeriodException
```

All Implemented Interfaces:
java.io.Serializable

Description
This exception is thrown when a specified period is invalid (for example, start time is after the end time)

Constructors

SIInvalidPeriodException()

```
public SIInvalidPeriodException()
```

SIInvalidPeriodException(String)

```
public SIInvalidPeriodException(java.lang.String reason)
```

Constructor for the exception with a specified reason

Parameters:
`reason` - the reason why the exception was raised

org.dvb.si
SIIterator

Syntax
```
public interface SIIterator extends java.util.Enumeration
```

All Superinterfaces:
java.util.Enumeration

Description
Objects implementing SIIterator interface allow to browse through a set of SI objects. In order to maintain consistency within the set of SI objects, this browsing does NOT initiate an actual access to the stream.

Methods

---

numberOfRemainingObjects()

```
public int numberOfRemainingObjects()
```

Get the number of remaining objects in the iterator.

Returns:
The number of remaining objects.

org.dvb.si
SILackOfResourcesEvent

Syntax
```
public class SILackOfResourcesEvent extends SIRetrievalEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--SIRetrievalEvent
              |
              +--org.dvb.si.SILackOfResourcesEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
This event is sent in response to a SI retrieval request when the resources needed for retrieving the data are not available, e.g. due to the necessary resources being all taken up by the calling application or other applications.

See Also:
SIRetrievalListener

Constructors

SILackOfResourcesEvent(Object, SIRequest)

```
public SILackOfResourcesEvent(java.lang.Object appData, SIRequest request)
```

The constructor for the event

Parameters:

`appData` - the application data passed in the request method call

`request` - the SIRequest instance which is the source of the event

org.dvb.si
SIMonitoringEvent

Syntax
```
public class SIMonitoringEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.si.SIMonitoringEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
Objects of this class are sent to listener objects of the using application to notify that a change in the monitored information has happened.

See Also:
SIMonitoringType, SIMonitoringListener

Constructors

SIMonitoringEvent(SIDatabase, byte, int, int, int, int, int, Date, Date)

```
public SIMonitoringEvent(SIDatabase source, byte objectType, int networkId, int bouquetId,
            int originalNetworkId, int transportStreamId, int serviceId,
            java.util.Date startTime, java.util.Date endTime)
```

Constructor for the event object

Parameters:

`SIDatabase` - the SIDatabase object which is the source of the event

`objectType` - type of the SIInformation object (constants in SIMonitoringType)

`networkId` - networkId

`bouquetId` - bouquetId

`originalNetworkId` - originalNetworkId

`transportStreamId` - transportStreamId

`serviceId` - serviceId

`startTime` - start time of event schedule period

`endTime` - end time of event schedule period

Methods

getBouquetID()

```
public int getBouquetID()
```

Returns the bouquetId of the bouquet. This method is only applicable if the SIInformation type returned with the getSIInformationType method is BOUQUET.

Returns:
the bouquetId or -2 if not applicable for this event

getEndTime()

```
public java.util.Date getEndTime()
```

Returns the end time of the schedule period whose event information has changed. This method is only applicable if the SIInformation type returned with the getSIInformationType method is SCHEDULED_EVENT.

Returns:
the end time or null if not applicable for this event

getNetworkID()

```
public int getNetworkID()
```

Returns the networkId of the network. This method is only applicable if the SIInformation type returned with the getSIInformationType method is NETWORK.

Returns:
the networkId or -2 if not applicable for this event

getOriginalNetworkID()

```
public int getOriginalNetworkID()
```

Returns the originalNetworkId of the SIInformation objects This method is only applicable if the SIInformation type returned with the getSIInformationType method is SERVICE, PMT_SERVICE, PRESENT_FOLLOWING_EVENT or SCHEDULED_EVENT.

Returns:
the originalNetworkId or -2 if not applicable for this event

getServiceID()

```
public int getServiceID()
```

Returns the serviceId of the SIInformation objects This method is only applicable if the SIInformation type returned with the getSIInformationType method is PMT_SERVICE, PRESENT_FOLLOWING_EVENT or SCHEDULED_EVENT.

Returns:
the serviceId or -2 if not applicable for this event

getSIInformationType()

```
public byte getSIInformationType()
```

Get the SIInformation type of the information that has changed

Returns:
The SIInformation type (the possible values are defined in the SIMonitoringType interface).

See Also:
SIMonitoringType

---

getSource()

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

Gets the SIDatabase instance that is sending the event.

Overrides:
java.util.EventObject.getSource() in class java.util.EventObject

Returns:
the SIDatabase instance that is the source of this event.

---

getStartTime()

```
public java.util.Date getStartTime()
```

Returns the start time of the schedule period whose event information has changed. This method is only applicable if the SIInformation type returned with the getSIInformationType method is SCHEDULED_EVENT.

Returns:
the start time or null if not applicable for this event

---

getTransportStreamID()

```
public int getTransportStreamID()
```

Returns the transportStreamId of the SIInformation objects This method is only applicable if the SIInformation type returned with the getSIInformationType method is SERVICE, PMT_SERVICE, PRESENT_FOLLOWING_EVENT or SCHEDULED_EVENT.

Returns:
the transportStreamId or -2 if not applicable for this event

org.dvb.si
SIMonitoringListener

Syntax
```
public interface SIMonitoringListener extends java.util.EventListener
```

All Superinterfaces:
java.util.EventListener

Description
This interface shall be implemented by using application classes in order to listen to changes in monitored SI objects.

See Also:
SIMonitoringEvent

Methods

---

postMonitoringEvent(SIMonitoringEvent)

```
public void postMonitoringEvent(SIMonitoringEvent anEvent)
```

This method is called back by the SI API implementation to notify the listener about an event.

Parameters:
`anEvent` - The notified event.

See Also:
SIMonitoringEvent

org.dvb.si
SIMonitoringType

Syntax
```
public interface SIMonitoringType
```

Description
This interface defines the constants corresponding to the SI information type values in SIMonitoringEvent.

See Also:
SIMonitoringListener, SIMonitoringEvent

Fields

BOUQUET
```
public static final byte BOUQUET
```
Constant for the type of SIInformation object: Bouquet

NETWORK
```
public static final byte NETWORK
```
Constant for the type of SIInformation object: Network

PMT_SERVICE
```
public static final byte PMT_SERVICE
```
Constant for the type of SIInformation object: PMTService

PRESENT_FOLLOWING_EVENT
```
public static final byte PRESENT_FOLLOWING_EVENT
```
Constant for the type of SIInformation object: Present or following event

SCHEDULED_EVENT
```
public static final byte SCHEDULED_EVENT
```
Constant for the type of SIInformation object: Scheduled event

SERVICE
```
public static final byte SERVICE
```
Constant for the type of SIInformation object: Service

org.dvb.si
SINetwork

Syntax
```
public interface SINetwork extends SIInformation
```

All Superinterfaces:
SIInformation

Description
This interface (together with the SITransportStreamNIT interface) represents a sub-table of the Network Information Table (NIT) describing a particular network.

Each object that implements the SINetwork interface is identified by the identifier network_id.

See Also:
SITransportStream, SITransportStreamNIT

Methods

---

getDescriptorTags()

```
public short[] getDescriptorTags()
```

This method defines extra semantics for the SIInformation.getDescriptorTags method. If the NIT sub-table on which this SINetwork object is based consists of multiple sections, then this method returns the descriptor tags in the order they appear when concatenating the descriptor loops of the different sections.

Overrides:
getDescriptorTags() in interface SIInformation

See Also:
SIInformation, getDescriptorTags()

---

getName()

```
public java.lang.String getName()
```

This method returns the name of this network. The name is extracted from the network_name_descriptor or optionally from the multilingual_network_name_descriptor. When this information is not available "" is returned. All control characters as defined in ETR 211 are ignored. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

Returns:
The network name of this network.

---

getNetworkID()

```
public int getNetworkID()
```

Get the identification of this network.

Returns:
The network identification identifier.

---

getShortNetworkName()

```
public java.lang.String getShortNetworkName()
```

This method returns the short name (ETR 211) of this network without emphasis marks. The name is extracted from the network_name_descriptor or optionally from the multilingual_network_name_descriptor. When this information is not available "" is returned. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

Returns:
The short network name of this network.

---

retrieveDescriptors(short, Object, SIRetrievalListener)

```
public SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener)
```

This method defines extra semantics for the SIInformation.retrieveDescriptors method (first prototype). If the NIT sub-table on which this SINetwork object is based consists of multiple sections, then this method returns the requested descriptors in the order they appear when concatenating the descriptor loops of the different sections.

Overrides:
retrieveDescriptors(short, Object, SIRetrievalListener) in interface SIInformation

Throws:
SIIllegalArgumentException

See Also:
SIInformation, retrieveDescriptors(short, Object, SIRetrievalListener)

---

retrieveDescriptors(short, Object, SIRetrievalListener, short[])

```
public SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, short[] someDescriptorTags)
```

This method defines extra semantics for the SIInformation.retrieveDescriptors method (second prototype). If the NIT sub-table on which this SINetwork object is based consists of multiple sections, then this method returns the requested descriptors in the order they appear when concatenating the descriptor loops of the different sections.

Overrides:
retrieveDescriptors(short, Object, SIRetrievalListener, short[]) in interface SIInformation

Throws:
SIIllegalArgumentException

See Also:
SIInformation, retrieveDescriptors(short, Object, SIRetrievalListener, short[])

---

retrieveSITransportStreams(short, Object, SIRetrievalListener, short[])

```
public SIRequest retrieveSITransportStreams(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, short[] someDescriptorTags)
```

Retrieve information associated with transport streams carried via the network.

The SIIterator that is returned with the event when the request completes successfully will contain one or more objects that implement the SITransportStreamNIT interface.

Parameters:

`retrievemode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptorTags is null, the application is not interested in descriptors. All non applicable tag values are ignored

Returns:

An SIRequest object

Throws:

SIIllegalArgumentException - thrown if the retrieveMode is invalid

See Also:

SIRequest, SIRetrievalListener, SITransportStreamNIT, DescriptorTag

org.dvb.si
SINotInCacheEvent

Syntax
```
public class SINotInCacheEvent extends SIRetrievalEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--SIRetrievalEvent
              |
              +--org.dvb.si.SINotInCacheEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
This event is sent in response to a SI retrieval request when the request was made with the FROM_CACHE_ONLY mode and the requested data is not present in the cache.

See Also:
SIRetrievalListener

Constructors

SINotInCacheEvent(Object, SIRequest)

```
public SINotInCacheEvent(java.lang.Object appData, SIRequest request)
```

The constructor for the event

Parameters:

`appData` - the application data passed in the request method call

`request` - the SIRequest instance which is the source of the event

org.dvb.si
SIObjectNotInTableEvent

Syntax
```
public class SIObjectNotInTableEvent extends SIRetrievalEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--SIRetrievalEvent
              |
              +--org.dvb.si.SIObjectNotInTableEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
This event is sent in response to a SI retrieval request when the SI table where the information about the requested object should be located has been retrieved but the requested object is not present in it. The reason may be that the object corresponding to the requested identifiers does not exist.

See Also:
SIRetrievalListener

Constructors

SIObjectNotInTableEvent(Object, SIRequest)

```
public SIObjectNotInTableEvent(java.lang.Object appData, SIRequest request)
```

The constructor for the event

Parameters:

`appData` - the application data passed in the request method call

`request` - the SIRequest instance which is the source of the event

org.dvb.si
SIRequest

Syntax
```
public class SIRequest

java.lang.Object
  |
  +--org.dvb.si.SIRequest
```

Description
Object instances of this class represent SI retrieval requests made by the application. The application may cancel the request using this object.

Methods

---

cancelRequest()

```
public boolean cancelRequest()
```

Cancels the retrieval request.

---

isAvailableInCache()

```
public boolean isAvailableInCache()
```

Returns whether the information will be returned from cache or from the stream

org.dvb.si
SIRequestCancelledEvent

Syntax
```
public class SIRequestCancelledEvent extends SIRetrievalEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--SIRetrievalEvent
              |
              +--org.dvb.si.SIRequestCancelledEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
This event is sent in response to a SI retrieval request when the request is cancelled with the SIRequest.cancelRequest method call.

See Also:
SIRequest, SIRetrievalListener

Constructors

SIRequestCancelledEvent(Object, SIRequest)

```
public SIRequestCancelledEvent(java.lang.Object appData, SIRequest request)
```

The constructor for the event

Parameters:

`appData` - the application data passed in the request method call

`request` - the SIRequest instance which is the source of the event

org.dvb.si
SIRetrievalEvent

Syntax
```
public abstract class SIRetrievalEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.si.SIRetrievalEvent
```

Direct Known Subclasses:
SILackOfResourcesEvent, SINotInCacheEvent, SIObjectNotInTableEvent, SIRequestCancelledEvent, SISuccessfulRetrieveEvent, SITableNotFoundEvent, SITableUpdatedEvent

All Implemented Interfaces:
java.io.Serializable

Description
This class is the base class for events about completion of a SI retrieval request. Exactly one event will be returned in response to an SI retrieval request.

See Also:
SIRetrievalListener

Constructors

SIRetrievalEvent(Object, SIRequest)

```
public SIRetrievalEvent(java.lang.Object appData, SIRequest request)
```

The constructor for the event

Parameters:

`appData` - the application data passed in the request method call

`request` - the SIRequest instance which is the source of the event

Methods

---

getAppData()

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

Returns the application data that was passed to the retrieve method

getSource()

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

Returns the SIRequest object that is the source of this event

Overrides:
java.util.EventObject.getSource() in class java.util.EventObject

Returns:
the SIRequest object

org.dvb.si
SIRetrievalListener

Syntax
```
public interface SIRetrievalListener extends java.util.EventListener
```

All Superinterfaces:
java.util.EventListener

Description
This interface shall be implemented by application classes in order to receive events about completion of SI requests.

See Also:
SIRetrievalEvent

Methods

---

postRetrievalEvent(SIRetrievalEvent)

```
public void postRetrievalEvent(SIRetrievalEvent event)
```

This method is called by the SI API implementation to notify the listener about completion of an SI request.

Parameters:
`event` - The event object.

See Also:
SIRetrievalEvent

org.dvb.si
SIRunningStatus

Syntax
```
public interface SIRunningStatus
```

Description
This interface defines the constants corresponding to the running status values for services and events.

Fields

NOT_RUNNING
```
public static final byte NOT_RUNNING
```

PAUSING
```
public static final byte PAUSING
```

RUNNING
```
public static final byte RUNNING
```

STARTS_IN_A_FEW_SECONDS
```
public static final byte STARTS_IN_A_FEW_SECONDS
```

UNDEFINED
```
public static final byte UNDEFINED
```

org.dvb.si
SIService

Syntax
```
public interface SIService extends SIInformation
```

All Superinterfaces:
SIInformation

Description
This interface represents a particular service carried by a transport stream. Information that can be obtained through the methods of this interface is retrieved from the SDT table.

Each object that implements the SIService interface is identified by the combination of the following identifiers: original_network_id, transport_stream_id, service_id.

Methods

---

getDvbLocator()

```
public org.davic.net.dvb.DvbLocator getDvbLocator()
```

Gets a DvbLocator that identifies this service.

Returns:
The DvbLocator of this service

---

getEITPresentFollowingFlag()

```
public boolean getEITPresentFollowingFlag()
```

Get the EIT_present_following_flag value, true indicates this service has present and/or following event information.

Returns:
The EIT_present_following_flag value.

---

getEITScheduleFlag()

```
public boolean getEITScheduleFlag()
```

Get the EIT_schedule_flag value, true indicates this services has scheduled event information.

Returns:
The EIT_schedule_flag value.

---

getFreeCAMode()

```
public boolean getFreeCAMode()
```

Retrieve the free_CA_mode value of this service, false indicates none of the components of this service are scrambled.

Returns:

The free_CA_mode value of this service.

---

getName()

```
public java.lang.String getName()
```

This method returns the name of the service represented by this service. The name is extracted from the service_descriptor or optionally from the multilingual_service_name_descriptor. If this descriptor is not present "" is returned. All control characters as defined in ETR 211 are ignored. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

Returns:
The name of this service.

---

getOriginalNetworkID()

```
public int getOriginalNetworkID()
```

Get the original network identification.

Returns:
The original network identification identifier.

---

getProviderName()

```
public java.lang.String getProviderName()
```

This method returns the service provider name of this service The service provider name is extracted from the service_descriptor or optionally from the multilingual_service_name_descriptor. If this descriptor is not present "" is returned. All control characters as defined in ETR 211 are ignored. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

Returns:
The service provider name of this service.

---

getRunningStatus()

```
public byte getRunningStatus()
```

Retrieve the running status of this service.

Returns:
The running status (the possible values are defined in the SIRunningStatus interface)

See Also:
SIRunningStatus

---

getServiceID()

```
public int getServiceID()
```

Get the service identification.

Returns:
The service identification identifier.

---

getShortProviderName()

```
public java.lang.String getShortProviderName()
```

This method returns the short name (ETR 211) of the service provider of this service without emphasis marks. The name is extracted from the service_descriptor or optionally from the multilingual_service_name_descriptor. When this information is not available "" is returned. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

Returns:
The short service provider name of this service.

---

getShortServiceName()

```
public java.lang.String getShortServiceName()
```

This method returns the short name (ETR 211) of this service without emphasis marks. The name is extracted from the service_descriptor or optionally from the multilingual_service_name_descriptor. When this information is not available "" is returned. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

Returns:
The short name of this service.

---

getSIServiceType()

```
public short getSIServiceType()
```

Get the service type. The service type is extracted from the service_descriptor.

Returns:
The service type. (Some of the possible values are defined in the SIServiceType interface.)

See Also:
SIServiceType

---

getTransportStreamID()

```
public int getTransportStreamID()
```

Get the transport stream identification.

Returns:
The transport stream identification identifier.

---

retrieveFollowingSIEvent(short, Object, SIRetrievalListener, short[])

```
public SIRequest retrieveFollowingSIEvent(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, short[] someDescriptorTags)
```

Retrieve information associated with the following event from the EIT-present/following.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SIEvent interface.

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 non applicable tag values are ignored.

Returns:
An SIRequest object

Throws:
SIIllegalArgumentException - thrown if the retrieveMode is invalid

See Also:
SIRequest, SIRetrievalListener, SIEvent, DescriptorTag

---

retrievePMTService(short, Object, SIRetrievalListener, short[])

```
public SIRequest retrievePMTService(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, short[] someDescriptorTags)
```

Retrieve the PMTService information associated with this service.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the PMTService interface.

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 non applicable tag values are ignored.

Returns:
An SIRequest object

Throws:
SIIllegalArgumentException - thrown if the retrieveMode is invalid

See Also:
SIRequest, SIRetrievalListener, PMTService, DescriptorTag

---

retrievePresentSIEvent(short, Object, SIRetrievalListener, short[])

```
public SIRequest retrievePresentSIEvent(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, short[] someDescriptorTags)
```

Retrieve information associated with the present event from the EIT-present/following.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SIEvent interface.

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 non applicable tag values are ignored.

Returns:
An SIRequest object

Throws:
SIIllegalArgumentException - thrown if the retrieveMode is invalid

See Also:
SIRequest, SIRetrievalListener, SIEvent, DescriptorTag

---

retrieveScheduledSIEvents(short, Object, SIRetrievalListener, short[], Date, Date)

```
public SIRequest retrieveScheduledSIEvents(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, short[] someDescriptorTags,
            java.util.Date startTime, java.util.Date endTime)
```

Retrieve information associated with the scheduled events within the service for a requested period from the EIT-schedule. The events are presented in the order they are present in the EIT-schedule.

The SIIterator that is returned with the event when the request completes successfully will contain one or more objects that implement the SIEvent interface.

Parameters:
`retrievemode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptorTags is null, the application is not interested in descriptors. All non applicable tag values are ignored.

`startTime` - The beginning of the required period in UTC time.

`endTime` - The end of the required period in UTC time.

Returns:
An SIRequest object

Throws:
SIIllegalArgumentException - thrown if the retrieveMode is invalid

SIInvalidPeriodException - When no valid period is indicated.

See Also:
SIRequest, SIRetrievalListener, SIEvent, DescriptorTag

org.dvb.si
SIServiceType

Syntax
```
public interface SIServiceType
```

Description
This interface defines constants corresponding to the different service types.

See Also:
getSIServiceType()

Fields

D_D2_MAC
```
public static final short D_D2_MAC
```

DATA_BROADCAST
```
public static final short DATA_BROADCAST
```

DIGITAL_RADIO_SOUND
```
public static final short DIGITAL_RADIO_SOUND
```

DIGITAL_TELEVISION
```
public static final short DIGITAL_TELEVISION
```

FM_RADIO
```
public static final short FM_RADIO
```

MHP_APPLICATION
```
public static final short MHP_APPLICATION
```

MOSAIC
```
public static final short MOSAIC
```

NTSC
```
public static final short NTSC
```

NVOD_REFERENCE

```
public static final short NVOD_REFERENCE
```

NVOD_TIME_SHIFTED

```
public static final short NVOD_TIME_SHIFTED
```

PAL

```
public static final short PAL
```

SECAM

```
public static final short SECAM
```

TELETEXT

```
public static final short TELETEXT
```

UNKNOWN

```
public static final short UNKNOWN
```

org.dvb.si
SISuccessfulRetrieveEvent

Syntax
```
public class SISuccessfulRetrieveEvent extends SIRetrievalEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--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. The returned SIIterator will always contain at least one object.

See Also:

SIObjectNotInTableEvent

org.dvb.si
SITableNotFoundEvent

Syntax
```
public class SITableNotFoundEvent extends SIRetrievalEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--SIRetrievalEvent
              |
              +--org.dvb.si.SITableNotFoundEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
This event is sent in response to a SI retrieval request when the SI table that should contain the requested information could not be retrieved. The reason may be that the requested table is not broadcast in the transport stream currently associated with the SI database.

See Also:
SIRetrievalListener

Constructors

SITableNotFoundEvent(Object, SIRequest)

```
public SITableNotFoundEvent(java.lang.Object appData, SIRequest request)
```

The constructor for the event

Parameters:

`appData` - the application data passed in the request method call

`request` - the SIRequest instance which is the source of the event

org.dvb.si
SITableUpdatedEvent

Syntax
```
public class SITableUpdatedEvent extends SIRetrievalEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--SIRetrievalEvent
              |
              +--org.dvb.si.SITableUpdatedEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
This event is sent in response to a SI descriptor retrieval request when the table carrying the information about the object has been updated and the set of descriptors consistent with the old object can not be retrieved. The application should in this case first update the SIInformation object and then request the descriptors again.

See Also:
SIRetrievalListener

Constructors

SITableUpdatedEvent(Object, SIRequest)

```
public SITableUpdatedEvent(java.lang.Object appData, SIRequest request)
```

The constructor for the event

Parameters:

`appData` - the application data passed in the request method call

`request` - the SIRequest instance which is the source of the event

org.dvb.si
SITime

Syntax
```
public interface SITime extends SIInformation
```

All Superinterfaces:
SIInformation

Description
This interface represents the Time and Date Table (TDT) and the (optional) Time Offset Table (TOT). When it represents a TDT table, the retrieveDescriptors and retrieveDescriptorTags methods will return empty SIIterators because no descriptors can be filtered from a TDT.

See Also:
SIDatabase

Methods

---

getUTCTime()

```
public java.util.Date getUTCTime()
```

Get the UTC time as coded in the TDT or TOT table.

Returns:
The UTC as coded in the TDT or TOT table.

org.dvb.si
SITransportStream

Syntax
```
public interface SITransportStream extends SIInformation
```

All Known Subinterfaces:
SITransportStreamBAT, SITransportStreamNIT

All Superinterfaces:
SIInformation

Description
This interface is the base interface for representing information about transport streams.

Transport stream retrieval methods in the SIDatabase class and the SINetwork interface use the NIT table and will return objects that implement the SITransportStreamNIT interface.

Transport stream retrieval methods in the SIBouquet interface use the BAT table and will return objects that implement the SITransportStreamBAT interface.

Methods

---

getDvbLocator()

```
public org.davic.net.dvb.DvbLocator getDvbLocator()
```

Gets a DvbLocator that identifies this transport stream.

Returns:
The DvbLocator of this transport stream.

---

getOriginalNetworkID()

```
public int getOriginalNetworkID()
```

Get the original network identification.

Returns:
The original network identification identifier.

---

getTransportStreamID()

```
public int getTransportStreamID()
```

Get the transport stream identification.

Returns:
The transport stream identification identifier.

retrieveSIServices(short, Object, SIRetrievalListener, short[])

```
public SIRequest retrieveSIServices(short retrieveMode, java.lang.Object appData,
            SIRetrievalListener listener, short[] someDescriptorTags)
```

Retrieve information associated with services carried via the transport stream. This method works in the same way for objects that implement the SITransportStreamNIT and SITransportStreamBAT interfaces.

The SIIterator that is returned with the event when the request completes successfully will contain objects that implement the SIService interface.

Parameters:

`retrievemode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptorTags is null, the application is not interested in descriptors. All non applicable tag values are ignored.

Returns:

An SIRequest object

Throws:

SIIllegalArgumentException - thrown if the retrieveMode is invalid

See Also:

SIRequest, SIRetrievalListener, SIService, DescriptorTag

org.dvb.si
SITransportStreamBAT

Syntax
```
public interface SITransportStreamBAT extends SITransportStream
```

All Superinterfaces:
SIInformation, SITransportStream

Description
This interface represents information about transport streams that has been retrieved from a BAT table. All descriptor accessing methods return descriptors retrieved from a BAT table. Methods in SIBouquet for retrieving transport streams return objects that implement this interface.

Methods

---

getBouquetID()

```
public int getBouquetID()
```

Get the identification of the bouquet this transport stream is part of.

Returns:
The bouquet identification identifier.

org.dvb.si
SITransportStreamDescription

Syntax
```
public interface SITransportStreamDescription extends SIInformation
```

All Superinterfaces:
SIInformation

Description
This interface represents the Transport Stream Description Table (TSDT).

It defines no methods of its own other than those inherited from SIInformation.

See Also:
SIDatabase, SITransportStream

org.dvb.si
SITransportStreamNIT

Syntax
```
public interface SITransportStreamNIT extends SITransportStream
```

All Superinterfaces:
SIInformation, SITransportStream

Description
This interface represents information about transport streams that has been retrieved from a NIT table. All descriptor accessing methods return descriptors retrieved from a NIT table. Methods in SIDatabase and SINetwork for retrieving transport streams return objects that implement this interface.

Methods

---

getNetworkID()

```
public int getNetworkID()
```

Get the identification of the network this transport stream is part of.

Returns:
The network identification identifier.

org.dvb.si
SIUtil

Syntax
```
public class SIUtil

java.lang.Object
  |
  +--org.dvb.si.SIUtil
```

Description
This class contains SI related utility functions.

Methods

---

convertSIStringToJavaString(byte[], int, int, boolean)

```
public static java.lang.String convertSIStringToJavaString(byte[] dvbSIText, int offset,
            int length, boolean emphasizedPartOnly)
```

This method converts a text string that is coded according to annex A of the DVB-SI specification (EN 300 468) to a Java String object.

The text that must be converted is contained in 'dvbSIText' from index 'offset' to index 'offset+length-1' (including).

If the text that must be converted is not validly coded according to annex A of the DVB-SI specification, then the result is undefined.

Parameters:

`dvbSIText` - The byte array that contains the string that must be converted.

`offset` - The offset indicates the start of the DVB-SI text in dvbSIText.

`length` - Length of the DVB-SI text in bytes.

`emphasizedPartOnly` - If emphasizedPartOnly is true, then only the text that is marked as emphasized (using the character emphasis on [0x86] and character emphasis off [0x87] control codes) will be returned. Otherwise, the character emphasis codes will be ignored, and all of the converted text will be returned.

Returns:

The converted text.

Throws:

SIIllegalArgumentException - thrown if offset and/or offset+length-1 is not a valid index in dvbSIText.

Annex N (normative): Streamed Media API Extensions

Package org.dvb.media

Class Summary

Interfaces

BackgroundVideoPresentationControl	A control to support the setting and querying of the video presentation for background players.
SubtitleListener	Report that a subtitle event has happened.
SubtitlingEventControl	Allow applications to register and unregister their interest in events related to the availability and presentation of subtitles.
VideoFormatControl	This class provides a means for applications to get information associated with the format and aspect ratio of the video being presented to the user.
VideoFormatListener	
VideoPresentationControl	A control to support setting and querying the video presentation.

Classes

ActiveFormatDescriptionChangedEvent	Event signalling that the transmitted active format definition has changed
AspectRatioChangedEvent	Event signalling that the aspect ratio of the transmitted video has changed
CAStopEvent	This event is generated whenever access to a service is withdrawn by the CA system, e.g.
DFCChangedEvent	Event signalling that the decoder format conversion being used has changed
DripFeedDataSource	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.
DripFeedPermission	This class represents a permission to access the drip feed mode.
PresentationChangedEvent	This event is generated whenever the content being presented by a player changes for reasons outside the control of the application.
SubtitleAvailableEvent	Report that subtitles are available to be presented having been unavailable.
SubtitleNotAvailableEvent	Inform an application that a subtitle stream has vanished from the network.
SubtitleNotSelectedEvent	Report that subtitles are not now selected.
SubtitleSelectedEvent	Report that subtitles are now selected.
VideoFormatEvent	The base class for all other events relating to changes in video format
VideoTransformation	VideoTransformation objects express video transformations, i.e.

Exceptions

CAException	This exception is thrown when access to a media stream is denied by the CA system.

org.dvb.media
ActiveFormatDescriptionChangedEvent

Syntax
```
public class ActiveFormatDescriptionChangedEvent extends VideoFormatEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--VideoFormatEvent
              |
              +--org.dvb.media.ActiveFormatDescriptionChangedEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
Event signalling that the transmitted active format definition has changed

Constructors

ActiveFormatDescriptionChangedEvent(Object, int)

```
public ActiveFormatDescriptionChangedEvent(java.lang.Object source, int newFormat)
```

Construct the event

Parameters:

`source` - the source of the event

`newFormat` - the new active format description

Methods

---

getNewFormat()

```
public int getNewFormat()
```

Get the new active format description

Returns:

the new active format description. The value of this is represented by one of the constants from the VideoFormatControl class

org.dvb.media
AspectRatioChangedEvent

Syntax
```
public class AspectRatioChangedEvent extends VideoFormatEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--VideoFormatEvent
              |
              +--org.dvb.media.AspectRatioChangedEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
Event signalling that the aspect ratio of the transmitted video has changed

Constructors

AspectRatioChangedEvent(Object, int)

```
public AspectRatioChangedEvent(java.lang.Object source, int newRatio)
```

Construct the event

Parameters:

`source` - the source of the event

`newRatio` - the new aspect ratio of the transmitted video

Methods

---

getNewRatio()

```
public int getNewRatio()
```

Get the new aspect ratio of the transmitted video

Returns:

the new aspect ratio of the video. The value of this is represented by one of the constants from the VideoFormatControl class

org.dvb.media

BackgroundVideoPresentationControl

Syntax

```
public interface BackgroundVideoPresentationControl extends VideoPresentationControl
```

All Superinterfaces:

javax.media.Control, VideoPresentationControl

Description

A control to support the setting and querying of the video presentation for background players.

Methods

---

getClosestMatch(VideoTransformation)

```
public VideoTransformation getClosestMatch(VideoTransformation t)
```

This method takes a video transformation and returns the closest match of that video transformation that can be supported for the currently selected video. If the input video transformation can be supported, then the output video transformation will have the same parameters as the input video transformation. The definition of 'closest match' is implementation dependent.

Parameters:

`t` - the input video transformation

Returns:

the closest match to the input video transformation. If the input video transformation is supported, then the input video transformation will be returned (the same instance), otherwise a newly created instance will be returned.

---

getVideoTransformation()

```
public VideoTransformation getVideoTransformation()
```

Returns:

the video transformation (clipping/scaling/positioning) that is currently used for displaying the video.

---

setVideoTransformation(VideoTransformation)

```
public boolean setVideoTransformation(VideoTransformation t)
```

Sets a new video transformation (clipping/scaling/positioning). If the new video transformation is not supported, then the video transformation will not be changed at all (no best effort attempt is made).

Parameters:

`t` - the new video transformation

Returns:

true if the video transformation is supported and has been set, false otherwise.

org.dvb.media
CAException

Syntax
```
public class CAException extends java.io.IOException

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--org.dvb.media.CAException
```

All Implemented Interfaces:
java.io.Serializable

Description
This exception is thrown when access to a media stream is denied by the CA system. It will typically be thrown by calls to DataSource.start() when access to the stream accessed by the DataSource is denied.
constructors

Constructors

CAException()

`public CAException()`

Constructor without a reason

CAException(String)

`public CAException(java.lang.String reason)`

Constructor with a reason

Parameters:

`reason` - the reason why access to the stream failed

org.dvb.media
CAStopEvent

Syntax
```
public class CAStopEvent extends javax.media.StopEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--javax.media.ControllerEvent
              |
              +--javax.media.TransitionEvent
                    |
                    +--javax.media.StopEvent
                          |
                          +--org.dvb.media.CAStopEvent
```

All Implemented Interfaces:
javax.media.MediaEvent, java.io.Serializable

Description
This event is generated whenever access to a service is withdrawn by the CA system, e.g. at the end of a free preview period. It is not generated when an attempt to construct a Player or DataSource fails due to CA restrictions, or when only some of the presented content is not available or alternate content is presented. Generation of this event informs the application that the Player is no longer presenting any content.

Constructors

CAStopEvent(Controller)

```
public CAStopEvent(javax.media.Controller source)
```

Construct an event.

Parameters:

`source` - the controller which was presenting the service

CAStopEvent(Controller, MediaLocator)

```
public CAStopEvent(javax.media.Controller source, javax.media.MediaLocator stream)
```

Construct an event.

Parameters:

`source` - the controller which was presenting the service

`stream` - the URL of the stream from which access has been withdrawn.

Methods

getStream()

```
public javax.media.MediaLocator getStream()
```

This method returns the stream from which access has been withdrawn.

org.dvb.media
DFCChangedEvent

Syntax
```
public class DFCChangedEvent extends VideoFormatEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--VideoFormatEvent
              |
              +--org.dvb.media.DFCChangedEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
Event signalling that the decoder format conversion being used has changed

Constructors

DFCChangedEvent(Object, int)

```
public DFCChangedEvent(java.lang.Object source, int newDFC)
```

Construct the event

Parameters:

`source` - the source of the event

`newDFC` - the new aspect ratio of the transmitted video

Methods

---

getNewDFC()

```
public int getNewDFC()
```

Get the new decoder format conversion

Returns:

the new decoder format conversion. The value of this is represented by one of the constants from the VideoFormatControl class

org.dvb.media
DripFeedDataSource

Syntax

```
public class DripFeedDataSource extends javax.media.protocol.PullDataSource

java.lang.Object
  |
  +--javax.media.protocol.DataSource
        |
        +--javax.media.protocol.PullDataSource
              |
              +--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 instanciate a player representing a MPEG-2 video decoder and have its source be a DripFeedDataSource instance.

A DripFeedDataSource instance can be obtained by calling the default constructor of the class.

A player that will be bound to a MPEG-2 video decoder (when realized) can be created with the following special URL (locator): "dripfeed://". It is also possible to use a decoder that was instanciated to play a broadcast MPEG-2 stream.

After having the DripFeedDataSource connected to a Player representing a MPEG-2 video decoder, the following rules applies:

- If the feed method is called when the player is in the "prefetched" state the image will be stored so that when the player goes in the "started" state it will be automatically displayed.

- If the feed method is called when the player is in the "started" mode, the frame shall be displayed immediately. In particular it is not required to feed a second frame to the decoder to display the first frame.

- If the feed method is called when the player is in any other state (or if the DripFeedDataSource is not connected to a player), it will be ignored by the application.

Constructors

DripFeedDataSource()

```
public DripFeedDataSource()
```

Constructor. A call to the constructor will throw a security exception if the application is not granted the right to use this mode.

Methods

connect()

```
public void connect()
```

This method shall not be used and has no effect. This source is considered as always connected.

Overrides:
javax.media.protocol.DataSource.connect() in class javax.media.protocol.DataSource

Throws:
`IOException`

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 garanteed.

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 synctactic 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

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

Returns:
the object that implements the control, or null.

getControls()

```
public java.lang.Object[] getControls()
```

Obtain the collection of objects that control the object that implements this interface. If no controls are supported, a zero length array is returned.

Overrides:
javax.media.protocol.DataSource.getControls() in class javax.media.protocol.DataSource

Returns:
the collection of object controls

getDuration()

```
public javax.media.Time getDuration()
```

This method shall not be used and has no effect.

Overrides:
javax.media.protocol.DataSource.getDuration() in class javax.media.protocol.DataSource

Returns:
DURATION_UNKNOWN.

getStreams()

```
public javax.media.protocol.PullSourceStream[] getStreams()
```

This method is not used and shall return null.

Overrides:
javax.media.protocol.PullDataSource.getStreams() in class javax.media.protocol.PullDataSource

start()

```
public void start()
```

This method shall not be used and has no effect. This source is considered as always started.

Overrides:
javax.media.protocol.DataSource.start() in class javax.media.protocol.DataSource

Throws:
`IOException`

stop()

```
public void stop()
```

This method shall not be used and has no effect. This source is considered as always started.

Overrides:
javax.media.protocol.DataSource.stop() in class javax.media.protocol.DataSource

Throws:
`IOException`

org.dvb.media
DripFeedPermission

Syntax
```
public class DripFeedPermission extends java.security.BasicPermission

java.lang.Object
  |
  +--java.security.Permission
        |
        +--java.security.BasicPermission
              |
              +--org.dvb.media.DripFeedPermission
```

All Implemented Interfaces:
java.security.Guard, java.io.Serializable

Description
This class represents a permission to access the drip feed mode.

Constructors

DripFeedPermission(String)

```
public DripFeedPermission(java.lang.String name)
```

Create a new DripFeedPermission. The parameter name is not used.

Parameters:

`name` - Name of the DripFeedPermission. Not used, shall be "".

DripFeedPermission(String, String)

```
public DripFeedPermission(java.lang.String name, java.lang.String actions)
```

Create a new DripFeedPermission. The parameters name and action are not used.

Parameters:

`name` - Name of the DripFeedPermission. Not used, shall be "".

`actions` - Not used, shall be "".

org.dvb.media
PresentationChangedEvent

Syntax
```
public class PresentationChangedEvent extends javax.media.ControllerEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--javax.media.ControllerEvent
              |
              +--org.dvb.media.PresentationChangedEvent
```

All Implemented Interfaces:
javax.media.MediaEvent, java.io.Serializable

Description
This event is generated whenever the content being presented by a player changes for reasons outside the control of the application. The state of the player does not change - only the content being presented.

Fields

CA_FAILURE
```
public static final int CA_FAILURE
```
Presentation changed due an action by the CA subsystem. Alternate content is being played, not the content selected by the user (e.g. adverts in place of a scrambled service)

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)
```
public PresentationChangedEvent(javax.media.Controller source)
```
Constructor for the event

Parameters:
`source` - the controller whose presentation changed

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 why access has been withdrawn.

Methods

getReason()

```
public int getReason()
```

This method returns the reason why access has been withdrawn.

getStream()

```
public javax.media.MediaLocator getStream()
```

This method returns the locator for the stream now being presented.

org.dvb.media
SubtitleAvailableEvent

Syntax
```
public class SubtitleAvailableEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.media.SubtitleAvailableEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
Report that subtitles are available to be presented having been unavailable.

Constructors

SubtitleAvailableEvent(SubtitlingLanguageControl)

```
public SubtitleAvailableEvent(org.davic.media.SubtitlingLanguageControl source)
```

Constructor.

Parameters:
`source` - the source of the event

Methods

---

getSource()

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

Return the control which was the source of the event.

Overrides:
java.util.EventObject.getSource() in class java.util.EventObject

Returns:
the source of the event

org.dvb.media
SubtitleListener

Syntax
```
public interface SubtitleListener extends java.util.EventListener
```

All Superinterfaces:
java.util.EventListener

Description
Report that a subtitle event has happened.

Methods

---

subtitleStatusChanged(EventObject)

```
public void subtitleStatusChanged(java.util.EventObject event)
```

Report a subtitle event has happened.

org.dvb.media
SubtitleNotAvailableEvent

Syntax
```
public class SubtitleNotAvailableEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.media.SubtitleNotAvailableEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
Inform an application that a subtitle stream has vanished from the network.

Constructors

SubtitleNotAvailableEvent(SubtitlingLanguageControl)

```
public SubtitleNotAvailableEvent(org.davic.media.SubtitlingLanguageControl source)
```

Constructor.

Parameters:
`source` - the source of the event

Methods

---

getSource()

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

Return the source of the event.

Overrides:
java.util.EventObject.getSource() in class java.util.EventObject

Returns:
the source of the event

org.dvb.media
SubtitleNotSelectedEvent

Syntax
```
public class SubtitleNotSelectedEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.media.SubtitleNotSelectedEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
Report that subtitles are not now selected. Even if subtitles are available in the network, they will not be presented.

Constructors

SubtitleNotSelectedEvent(SubtitlingLanguageControl)

```
public SubtitleNotSelectedEvent(org.davic.media.SubtitlingLanguageControl source)
```

Constructor

Parameters:
`source` - the source of the event

Methods

---

getSource()

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

Return the source of the event

Overrides:
java.util.EventObject.getSource() in class java.util.EventObject

Returns:
the source of the event

org.dvb.media
SubtitleSelectedEvent

Syntax
```
public class SubtitleSelectedEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.media.SubtitleSelectedEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
Report that subtitles are now selected. If subtitles are also available then they will be presented.

Constructors

SubtitleSelectedEvent(SubtitlingLanguageControl)

```
public SubtitleSelectedEvent(org.davic.media.SubtitlingLanguageControl source)
```

Constructor

Parameters:
`source` - the source of the event

Methods

---

getSource()

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

Return the source of the event

Overrides:
java.util.EventObject.getSource() in class java.util.EventObject

Returns:
the source of the event

org.dvb.media
SubtitlingEventControl

Syntax
```
public interface SubtitlingEventControl extends org.davic.media.SubtitlingLanguageControl
```

All Superinterfaces:
javax.media.Control, org.davic.media.LanguageControl, org.davic.media.SubtitlingLanguageControl

Description
Allow applications to register and unregister their interest in events related to the availability and presentation of subtitles.

Methods

---

addSubtitleListener(SubtitleListener)

```
public void addSubtitleListener(SubtitleListener l)
```

Add a listener for subtitle events

Parameters:
`l` - the listener to report the events to

---

removeSubtitleListener(SubtitleListener)

```
public void removeSubtitleListener(SubtitleListener l)
```

Remove a listener for subtitle events

Parameters:
`l` - the listener to remove

org.dvb.media
VideoFormatControl

Syntax
```
public interface VideoFormatControl extends javax.media.Control
```

All Superinterfaces:
javax.media.Control

Description
This class 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_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, or the video is not MPEG, 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

getDecoderFormatConversion()

```
public int getDecoderFormatConversion()
```

Return a value representing what format conversion is being done by the decoder in the platform (one of the constants DFC_* above). A receiver may implement only a subset of the available options. This decoder format conversion may be active or not depending upon the mode of operation.

Returns:

the decoder format conversion being performed or DFC_PROCESSING_UNKNOWN if this is not known

---

getDisplayAspectRatio()

```
public int getDisplayAspectRatio()
```

Return the aspect ratio of the display device connected to this MHP decoder (one of the constants DAR_* above)

Returns:
the aspect ratio of the display device connected to the decoder

---

getVideoTransformation(int)

```
public VideoTransformation getVideoTransformation(int dfc)
```

This method returns a VideoTransformation object that corresponds with the specified Decoder Format Conversion when applied to the currently selected video. If the specified Decoder Format Conversion is not supported for the currently selected video, then this method returns null.

Parameters:
`dfc` - the Decoder Format Conversion (one of the DFC_* constants specified in this interface)

Returns:
the video transformation, or null if the specified Decoder Format Conversion is not supported for the currently selected video.

---

removeVideoFormatListener(VideoFormatListener)

```
public void removeVideoFormatListener(VideoFormatListener l)
```

Remove a listener for VideoFormatChangedEvents

Parameters:
`l` - the listener to remove

org.dvb.media
VideoFormatEvent

Syntax
```
public abstract class VideoFormatEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.media.VideoFormatEvent
```

Direct Known Subclasses:
ActiveFormatDescriptionChangedEvent, AspectRatioChangedEvent, DFCChangedEvent

All Implemented Interfaces:
java.io.Serializable

Description
The base class for all other events relating to changes in video format

Constructors

VideoFormatEvent(Object)

```
public VideoFormatEvent(java.lang.Object source)
```

Constructor

Parameters:

`source` - the source of the event

org.dvb.media
VideoFormatListener

Syntax
```
public interface VideoFormatListener
```

Methods

---

receiveVideoFormatEvent(VideoFormatEvent)

```
public void receiveVideoFormatEvent(VideoFormatEvent anEvent)
```

receive a VideoFormatEvent

Parameters:

`anEvent` - the VideoFormatEvent that has been received

org.dvb.media
VideoPresentationControl

Syntax
```
public interface VideoPresentationControl extends javax.media.Control
```

All Known Subinterfaces:
BackgroundVideoPresentationControl

All Superinterfaces:
javax.media.Control

Description
A control to support setting and querying the video presentation.

Note: For a component-based player the scaling and positioning of the video is done by manipulating the corresponding AWT component. The VideoPresentationControl only allows for the setting of the clipping region.

Note: If the hardware supports the positioning of interlaced video on even lines only (when counting from 0), then a component-based player is allowed to position the top of the video one line below where it should be.

For a background player there is the BackgroundVideoPresentationControl that allows for the setting of the clipping region, the position and the scaling of the video in one atomic action.

Fields

POS_CAP_FULL
```
public static final byte POS_CAP_FULL
```
Constant representing that the video can be positioned anywhere on the screen, even if a part of the video is off screen as a result of that.

POS_CAP_FULL_EVEN_LINES
```
public static final byte POS_CAP_FULL_EVEN_LINES
```
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 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 HScreenRectangle getActiveVideoAreaOnScreen()
```

This method returns the size and location of the active video area on-screen. The active video area excludes any "bars" used for letterboxing or pillarboxing that the receiver knows about. Bars that are included in the broadcast stream and not signalled by active format descriptors are included in the active video area. The active video area on-screen may be smaller than the area of the screen, and may possibly be offset a positive amount. This method only describes the area on-screen where active video is being presented. It does not really describe which part of the video is being shown on-screen. This is especially true for pan&scan.

Note: This method includes any video scaling.

Returns:
an HScreenRectangle representing the active video area on-screen in the normalised coordinate space.

getClipRegion()

```
public java.awt.Rectangle getClipRegion()
```

This method returns the area of the decoded video that will be displayed. If clipping is not supported, the dimensions of the bounding box will be the same as the displayed video.

Returns:

area of the decoded video that will be displayed. The coordinate space used to express the region is that of the decoded video after possible ETR154 up-sampling.

---

getHorizontalScalingFactors()

```
public float[] getHorizontalScalingFactors()
```

This method gives information about the supported discrete horizontal scaling factors in case arbitrary horizontal scaling is not supported.

Returns:

an array with the supported discrete horizontal scaling factors (including the scaling factor 1), sorted in ascending order. null is returned when arbitrary horizontal scaling is supported.

---

getInputVideoSize()

```
public java.awt.Dimension getInputVideoSize()
```

This method returns the dimensions of the video before any scaling has taken place (but after ETR154 up-sampling). On 50Hz standard definition systems this method always returns 720x576.

Returns:

the size of the decoded video before any scaling has taken place (but after ETR154 up-sampling)

---

getPositioningCapability()

```
public byte getPositioningCapability()
```

This method gives information about how the video can be positioned on screen.

Returns:

the positioning capability for the currently selected video as one of the POS_CAP_* constants.

---

getTotalVideoArea()

```
public 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 HScreenRectangle getTotalVideoAreaOnScreen()
```

This method returns a relative size and location of the total video area on-screen, including any "bars" used for letterboxing or pillarboxing that are included in the broadcast stream, but excluding any "bars" introduced as a result of video filtering. This method only describes the area on-screen

where total video is being presented. This does not really describe which part of the video is being shown on-screen. This is especially true for pan&scan.

Note: This method includes any video scaling.

Returns:
an HScreenRectangle representing the total video area on-screen in the normalised coordinate space.

---

getVerticalScalingFactors()

```
public float[] getVerticalScalingFactors()
```

This method gives information about the supported discrete vertical scaling factors in case arbitrary vertical scaling is not supported.

Returns:
an array with the supported discrete vertical scaling factors (including the scaling factor 1), sorted in ascending order. null is returned when arbitrary vertical scaling is supported.

---

getVideoSize()

```
public java.awt.Dimension getVideoSize()
```

This method returns the size of the decoded video as it is being presented to the user. It takes scaling and clipping into account.

Returns:
the size of the decoded video as it is being presented to the user

---

setClipRegion(Rectangle)

```
public java.awt.Rectangle setClipRegion(java.awt.Rectangle clipRect)
```

Set the region of the decoded video that will be displayed. If clipping is not supported, this method has no effect. If the bounding box extends beyond the decoded video, the area not containing video will be filled with the background. By default, the clipping region is set to the dimensions of the decoded video. This method returns the bounding box of the clipping region that was actually set. 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, 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 in 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 sup-

ported, 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 in 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()
```

This method returns true if and only if the decoder supports displaying only a section of the decoded video.

org.dvb.media
VideoTransformation

Syntax

```
public class VideoTransformation

java.lang.Object
  |
  +--org.dvb.media.VideoTransformation
```

Description

VideoTransformation objects express video transformations, i.e. the clipping, the horizontal and vertical scaling and the position of the video. All transformations are to be applied after possible ETR154 up-sampling.

Note: Instances of VideoTransformation can represent pan and scan, but an application cannot create such instances itself. An application can get a VideoTransformation representing pan and scan, by calling the VideoFormatControl.getVideoTransformation() method with the pan and scan Decoder Format Conversion constant.

Constructors

VideoTransformation()

```
public VideoTransformation()
```

Creates a VideoTransformation object with default parameters. Clipping is disabled, both the horizontal and the vertical scaling factors are 1, and the video position is (0,0) in the normalised coordinate space.

VideoTransformation(Rectangle, float, float, HScreenPoint)

```
public VideoTransformation(java.awt.Rectangle clipRect, float horizontalScalingFactor,
            float verticalScalingFactor, HScreenPoint location)
```

Creates a VideoTransformation object with the supplied parameters.

Parameters:

`clipRect` - the bounding box of the clipping region. The coordinate space used to express the region is that of the decoded video after possible ETR154 up-sampling.

`horizontalScalingFactor` - the horizontal scaling factor.

`verticalScalingFactor` - the vertical scaling factor.

`location` - the location of the video on the screen in the normalised coordinate space.

Methods

---

getClipRegion()

```
public java.awt.Rectangle getClipRegion()
```

Gets the clipping region.

Returns:
the bounding box of the clipping region. The coordinate space used to express the region is that of the decoded video after possible ETR154 up-sampling. null is returned if this video transformation represents pan and scan.

---

getScalingFactors()

```
public float[] getScalingFactors()
```

Gets the horizontal and vertical scaling factors.

Returns:
an array with two elements. The first element contains the horizontal scaling factor, the second element the vertical scaling factor.

---

getVideoPosition()

```
public 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(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 [53] 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 262) 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.

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 from of the service name from the Service descriptor.

O.2.1.2 getServiceType

Returns the ServiceType according to the mapping defined in section O.2.3, "javax.tv.service.ServiceType" on page 378.

O.2.2 javax.tv.service.ServiceComponent

The ServiceComponent interface provides the information contained in the Component descriptors, Multilingual component descriptors or Data broadcast descriptors in the EIT.

O.2.2.1 getComponentName

Returns the component description from the Component descriptor or from the Multilingual component descriptor using the language preference, or from the Data broadcast descriptor.

O.2.2.2 getAssociatedLanguage

Returns the ISO 639 language code from the Component descriptor or from the Data broadcast descriptor.

O.2.2.3 getStreamType

Returns the stream type according to the mapping from the stream_content field and the component_type field of the Component descriptor or the Data broadcast descriptor to the JavaTV stream types according to section

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.ServiceInformationFormat

This interface is implemented by objects implementing the Network, Bouquet, TransportStream, ServiceDetails, Service-Component and ProgramEvent interfaces.

O.2.5.1 getServiceInformationType

This method shall return the DVB_SI ServiceInformationType.

O.2.6 javax.tv.service.navigation.SIManager

O.2.6.1 getSupportedDimensions

The parental rating descriptor defined in DVB SI standardizes one rating scheme that is based on age. To describe this DVB defined rating scheme, the getSupportedDimensions shall return an array that contains the string "DVB Age based rating".

O.2.6.2 getRatingDimension

When given the string "DVB Age based rating", this method shall return an object implementing the RatingDimension interface as described in section O.2.10, "javax.tv.service.navigation.RatingDimension" on page 380.

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 380.

O.2.6.5 createServiceCollection

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 379

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 380. These objects shall not implement the ServiceNumber interface.

O.2.8.1 getLongName

Returns the full name of the service from the Service descriptor or from the Multilingual service name descriptor using the language preference.

O.2.8.2 getServiceType

Returns the ServiceType according to the mapping defined in section O.2.3, "javax.tv.service.ServiceType" on page 378.

O.2.8.3 retrieveServiceDescription

Shall always result in a notifyFailure of the SIRequestor object being called with the DATA_UNAVAILABLE SIRequestFailureType, as DVB SI does not include a service description.

O.2.8.4 retrieveComponents

The information for the ServiceComponents shall be retrieved from Component descriptors (or their multilingual variants or the Data broadcast descriptor for data components) of the present event in the EIT present/following table.

O.2.9 javax.tv.service.navigation.CAIdentification

This interface shall be implemented by objects implementing the ServiceDetails, ProgramEvent or Bouquet interface.

O.2.9.1 getCASystemIds

Returns the array of integer values containing the CA_system_ids from the CA identifier descriptor. If the CA identifier descriptor is not present, returns an empty array.

O.2.9.2 isFree

When implemented in an object implementing the ServiceDetails or ProgramEvent interface, this method shall return true if and only if the free_CA_mode bit is set to "0" in the SDT or EIT entry, respectively.

When implemented in an object implementing the Bouquet interface, this method shall return true if and only if there is no CA identifier descriptor present in the BAT.

O.2.10 javax.tv.service.navigation.RatingDimension

The Parental rating descriptor defined in DVB SI standardizes one rating scheme that is based on age. This rating scheme contains 15 distinct age rating levels from 4 to 18 years.

An object that describes this DVB defined rating scheme shall implement the methods as follows.

O.2.10.1 getDimensionName

Returns the string "DVB Age based rating".

O.2.10.2 getNumberOfLevels

Returns 15.

O.2.10.3 getRatingLevelDescription

Returns an array of 2 strings of the form:

{"Over *n*", "Recommended minimum age: *n* years"}

where n is the input parameter + 4.

O.2.11 javax.tv.service.navigation.ServiceProviderInformation

This interface shall be implemented by objects implementing the ServiceDetails interface.

O.2.11.1 getProviderName

Returns the service provider name from the Service descriptor.

O.2.12 javax.tv.service.transport.Transport

The object implementing the Transport interface shall also implement the interfaces NetworkCollection and Bouquet-Collection.

O.2.13 javax.tv.service.transport.Bouquet

The Bouquet interface is implemented by an object that represents a DVB SI Bouquet.

O.2.13.1 getBouquetID

Returns the integer DVB SI Bouquet ID value.

O.2.13.2 getName

Returns the name of the bouquet from the Bouquet name descriptor or from the Multilingual bouquet name descriptor using the language preference.

O.2.13.3 getLocator

Returns an implementation dependent javax.tv.locator.Locator object that does not have a standardized external representation and might not be a org.davic.net.dvb.DvbLocator.

O.2.14 javax.tv.service.transport.Network

The Network interface is implemented by an object that represents a DVB SI Network.

O.2.14.1 getNetworkID

Returns the integer DVB SI Network ID value.

O.2.14.2 getName

Returns the name of the network from the Network name descriptor or from the Multilingual network name descriptor using the language preference.

O.2.14.3 getLocator

Returns an implementation dependent javax.tv.locator.Locator object that does not have a standardized external representation and might not be a org.davic.net.dvb.DvbLocator.

O.2.15 javax.tv.service.transport.TransportStream

The TransportStream interface is implemented by an object that represents a transport stream.

O.2.15.1 getTransportStreamID

Returns the integer DVB SI transport stream ID value.

O.2.15.2 getDescription

Transport streams do not have descriptions in DVB SI, so this method shall return an empty string.

O.2.16 javax.tv.service.guide.ProgramEvent

This interface is implemented by objects representing DVB SI Events.

O.2.16.1 getDuration

Returns the duration value from the event entry in the body of the EIT.

O.2.16.2 getStartTime

Returns the start time value from the event entry in the body of the EIT.

O.2.16.3 getEndTime

Returns the end time value calculated from the start time and duration in the body of the EIT.

O.2.16.4 getName

Returns the event name from one of the Short event descriptors in the event using the language preference.

O.2.16.5 retrieveDescription

Returns the event description from one of the Short event descriptors in the event using the language preference.

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 382.

O.2.17 javax.tv.service.guide.ContentRatingAdvisory

O.2.17.1 getDimensionNames

Returns an array containing the string "DVB Age based rating".

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 -1 from the parental rating descriptor. Otherwise 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". Otherwise returns an empty string.

O.2.17.4 getDisplayText

When the parental rating descriptor contains a rating value between 0x01 and 0x0F for the current region, returns a string that contains the string "Over n" as its substring.

O.3 Integration of the JavaTV SI API and the DVB SI API

In order for the protocol independent service information API to be useful, there needs to be an easy and convenient way for applications to use the DVB specific parts when they are needed. The information provided in the protocol independent API is quite minimal and does not cover all the aspects of the standardized DVB Service Information nor access to the private extensions carried in the standard protocol. If there is no integration between these APIs and the application programmer needs to use a completely different API to retrieve additional information on the object retrieved from the protocol independent API, the usefulness of the protocol independent API is very questionable. In this case, the application programmer will start using only the protocol dependent API, as it provides the complete information and is as easy to use as the other API.

To overcome these problems and make the protocol independent API somehow useful, it needs to be well integrated with the protocol dependent API, so that if an application uses first the protocol independent API for browsing the information, it can easily get additional, protocol dependent information on the objects of interest.

The Java language provides an easy way to achieve this integration: the same objects can implement both the protocol independent interface as well as the protocol dependent interface. This way the application programmer only needs to cast the object to the protocol dependent interface and can directly call methods from the protocol specific API.

Objects implementing the following interfaces of the DVB SI API should implement also the corresponding JavaTV SI API interfaces. When retrieving SI objects through the JavaTV APIs, they shall also implement the corresponding DVB SI API interfaces.

The interfaces of both APIs shall be implemented on the objects as follows:

org.dvb.si.SINetwork	objects implement also	javax.tv.service.transport.Network
org.dvb.si.SIBouquet	objects implement also	javax.tv.service.transport.Bouquet
org.dvb.si.SITransportStreamNIT	objects implement also	javax.tv.service.transport.TransportStream
org.dvb.si.SIService	objects implement also	javax.tv.service.navigation.ServiceDetails
org.dvb.si.SIEvent	objects implement also	javax.tv.service.guide.ProgramEvent

Annex P (normative): Broadcast Transport Protocol Access

The Object Carousel represents the best suited protocol to carry a structure of objects. Thus, the Object Carousel 'mimics' a remote server.

The structure of the objects carried in an Object Carousel is identical to the structure of UU-Objects located on a remote DSMCC-UU Server.

The aim of this API is to enable an application to access files encapsulated in an object carousel or accessible through a DSMCC interactive network. Note that the protocol is abstracted from the application viewpoint, so, objects accessible through this API are either objects encapsulated in an Object Carousel, or Objects located in an interactive DSMCC network on a remote server.

To benefit from the fact that most of the functionalities are already covered by the java.io package, this API inherits from java.io and only defines the extra-functionalities pertaining to:

a) the nature of the network (broadcast or DSMCC remote server) and its latency (e.g. possibility to asynchronously load the objects)

b) the type of the objects that can be encapsulated in a carousel and that do not exist in a classical File structure. These are: ServiceGateway, Directory, File, Stream and StreamEvent.

c) Definition of ServiceGateway, which defines a new namespace corresponding to the new Domain, and enables the mounting of a new volume.

An application can optionally use only the classes of java.io. Alternatively/additionally applications can use additional classes and methods adapted to the specific nature and latency of the network (such as for example, the asynchronous loading of objects).

The following, briefly explains the functionalities offered by this API

The ServiceDomain class enables attaching to a ServiceDomain. Attachment to a serviceDomain corresponds to the mounting of a volume in the file hierarchy system and the loading of the Service Gateway.

When attached to a Service Domain the DSM-CC UU-File, UU-Stream, UU-Directory and UU-StreamEvent objects are accessible through this API.

The class DSMCCObject represents a UU-object. Due to the close relationship between resident files and downloaded files, this class inherits from the java.io.File class. The DSMCCObject class just defines the additional methods specific to DSMCC-UU that basically deal with asynchronous or synchronous loading of Objects.

For the UU-Files or UU-Directory Objects, their content is accessible as it would be for a classical file system, that means through the classical JDK java.io package (e.g., for listing the objects pointed to by a Directory object, you invoke the list() method of the java.io.File class, or to access the content of a UU-File, you can instantiate a FileInputStream to read the File, etc…).

Additionally, the DSMCCStream and StreamEvent 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 inherits from the DSMCCObject class and provides access to the following attributes Duration, Description, current NPT. In addition, an application can retrieve the list of Taps (modelized 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 ObjectEvent class represents an event that is 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 ObjectEvent listeners are interfaces that must be implemented by the application, in order for it to receive the respective StreamEvents and ObjectChangeEvents.

Package org.dvb.dsmcc

Class Summary

Interfaces

AsynchronousLoadingEventListener	Listener for applications which perform asynchronous loading, in order to be informed if the loading is done or if an error has occured.
ObjectChangeEventListener	The objects that implements the ObjectChangeEventListener interface can receive ObjectChangeEvent event.
StreamEventListener	The objects that implements the StreamEventListener interface can receive StreamEvent event.

Classes

AsynchronousLoadingEvent	This class described an Object event which is used to notify the loading of a DSMCC object.
DSMCCObject	A DSMCCObject is an object which belongs to a DSMCC ServiceDomain.
DSMCCStream	The Stream class is used to manage DSMCC Stream Objects.
DSMCCStreamEvent	The DSMCCStreamEvent class is used to manage DSMCC StreamEvent Objects.
InvalidFormatEvent	The format of the data received is inconsistent
InvalidPathnameEvent	The pathname doe not exist or the ServiceDomain has been detached.
MPEGDeliveryErrorEvent	Error (for instance, a time out or accessing the data would require tuning) has occured while loading data from an MPEG Stream.
NotEntitledEvent	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.
ObjectChangeEvent	This class describes an ObjectChange event that is used to monitor the arrival of a new version of a DSMCC Object in an Object Carousel.
ServerDeliveryErrorEvent	The local machine can not communicate with the server.
ServiceDomain	A ServiceDomain is a group of DSMCC object.
ServiceXFRErrorEvent	The object requested is available in an alternate ServiceDomain.
ServiceXFRReference	A ServiceXFRReference object is used when a DSMCC Object can not be loaded in the current ServiceDomain but is available in an alternate ServiceDomain.
StreamEvent	This class describes a Stream event which is used to synchronize an application with an MPEG Stream.
SuccessEvent	This event indicates that the asynchronous loading was successful.

Exceptions

DSMCCException	The DSMCCException is the root class of all DSMCC related exceptions
IllegalObjectTypeException	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

Class Summary	
InvalidAddressException	A InvalidAddressException is thrown when the format of an NSAP address is not recognized.
InvalidFormatException	An InvalidFormatException is thrown when an inconsistent DSMCC message is received.
InvalidPathNameException	The InvalidPathNameException is thrown when the PathName to a DSMCCObject does not exist or if the ServiceDomain has been detached.
MPEGDeliveryException	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.
NotEntitledException	A 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)
NothingToAbortException	A NothingToAbortException is thrown when the abort method is called and there is no loading in progress.
NotLoadedException	A NotLoadedException is thrown when the Stream object constructor is called with a DSMCC Object which is not loaded.
ServerDeliveryException	A ServerDeliveryException is thrown when the local machine can not communicate with the server.
ServiceXFRException	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.
UnknownEventException	The UnknownEventException is thrown when a method tries to access to an unknown event.

org.dvb.dsmcc
AsynchronousLoadingEvent

Syntax
```
public abstract class AsynchronousLoadingEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.dsmcc.AsynchronousLoadingEvent
```

Direct Known Subclasses:
InvalidFormatEvent, InvalidPathnameEvent, MPEGDeliveryErrorEvent, NotEntitledEvent, ServerDeliveryErrorEvent, ServiceXFRErrorEvent, SuccessEvent

All Implemented Interfaces:
java.io.Serializable

Description
This class described an Object event which is used to notify the loading of a DSMCC object.

Constructors

AsynchronousLoadingEvent(DSMCCObject)

```
public AsynchronousLoadingEvent(DSMCCObject o)
```

The constructor for this 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

org.dvb.dsmcc
AsynchronousLoadingEventListener

Syntax
```
public interface AsynchronousLoadingEventListener extends java.util.EventListener
```

All Superinterfaces:
java.util.EventListener

Description
Listener for applications which perform asynchronous loading, in order to be informed if the loading is done or if an error has occured.

Methods

---

receiveEvent(AsynchronousLoadingEvent)

```
public void receiveEvent(AsynchronousLoadingEvent e)
```

Method called when an event is sent to the application.

Parameters:
`e` - an AsynchronousLoadingEvent event.

org.dvb.dsmcc
DSMCCException

Syntax
```
public class DSMCCException extends java.io.IOException

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--org.dvb.dsmcc.DSMCCException
```

Direct Known Subclasses:
IllegalObjectTypeException, InvalidAddressException, InvalidFormatException, InvalidPathNameException, MPEGDeliveryException, NotEntitledException, NothingToAbortException, NotLoadedException, ServerDeliveryException, ServiceXFRException, UnknownEventException

All Implemented Interfaces:
java.io.Serializable

Description
The DSMCCException is the root class of all DSMCC related exceptions

Constructors

DSMCCException()

```
public DSMCCException()
```

Construct a DSMCCException with no detail message

DSMCCException(String)

```
public DSMCCException(java.lang.String s)
```

Construct a DSMCCException with the specified detail message

Parameters:
`s` - the detail message

org.dvb.dsmcc
DSMCCObject

Syntax
```
public class DSMCCObject extends java.io.File

java.lang.Object
  |
  +--java.io.File
        |
        +--org.dvb.dsmcc.DSMCCObject
```

All Implemented Interfaces:
java.io.Serializable

Description
A DSMCCObject is an object which belongs to a DSMCC ServiceDomain. As soon as a ServiceDomain has been attached to the filesystem hierarchy, DSMCCObject objects can be created to access the ServiceDomain objects. To create a DSMCCObject the pathName has to begin with the volumeName associated with the ServiceDomain. 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: - It is allowed to call the constructor of java.io.FileInputStream (or java.io.RandomAccessFile) if the DSMCCObject is not loaded. But all the methods of FileInputStream (or RandomAccessFile) will throw an IOException until the Object is loaded. - Obviously, for the Object Carousel, the write mode of java.io.RandomAccessFile is not allowed.

See Also:
ServiceDomain

Constructors

DSMCCObject(DSMCCObject, String)

```
public DSMCCObject(DSMCCObject dir, java.lang.String name)
```

Create a DSMCCObject object.

Parameters:

`dir` - the directory object.

`name` - the filename.

DSMCCObject(Locator)

```
public DSMCCObject(org.davic.net.Locator locator)
```

Create a DSMCCObject object.

Parameters:

`locator` - A locator referencing the source of the DSMCCObject.

DSMCCObject(String)

```
public DSMCCObject(java.lang.String path)
```

Create a DSMCCObject object.

Parameters:

`path` - the FilePath.

DSMCCObject(String, String)

```
public DSMCCObject(java.lang.String path, java.lang.String name)
```

Create a DSMCCObject object.

Parameters:

`path` - the directory Path.

`name` - the filename.

Methods

abort()

```
public void abort()
```

This function 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.

addObjectChangeEventListener(ObjectChangeEventListener)

```
public void addObjectChangeEventListener(ObjectChangeEventListener listener)
```

Subscribes an ObjectChangeEventListener to receive notifications of version changes of DSMC-CObject.

Parameters:

`listener` - the ObjectChangeEventListener to be notified .

asynchronousLoad(AsynchronousLoadingEventListener)

```
public void asynchronousLoad(AsynchronousLoadingEventListener l)
```

This function is used to asynchronously load a carousel object. 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 occured: SuccessEvent, InvalidFormatEvent, InvalidPathNameEvent, MPEGDeliveryErrorEvent, ServerDeliveryErrorEvent, ServiceXFRErrorEvent, NotEntitledEvent

Parameters:

`l` - a listener to receive events related to asynchronous loading.

Throws:

DSMCCException - there is already a loading on this object.

InvalidPathNameException - The object can not be found.

getLocator()

```
public org.davic.net.Locator getLocator()
```

Returns a Locator identifying this carousel object.

Returns:
a Locator identifying the carousel object.

getVolumeName()

```
public java.lang.String getVolumeName()
```

This function is used to get the name of the volume where the carousel object is located. The volume name returned is the name that was given to the ServiceDomain constructor.

Returns:
the name of the volume where the carousel object is located.

isLoaded()

```
public boolean isLoaded()
```

Returns a boolean indicating whether or not the DSMCCObject has been loaded.

Returns:
true if the file is already loaded.

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 containg it is loaded).

Returns:
true if the type of the object is known.

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 streamEvent, false if the object is not a streamEvent or if the object kind is unknown.

loadDirectoryEntry(AsynchronousLoadingEventListener)

```
public void loadDirectoryEntry(AsynchronousLoadingEventListener l)
```

Asynchronous loading of the directory entry infrmation.

Parameters:

`l` - a listener which will be called when the loading is done.

Throws:

DSMCCException - if the directory is already being loaded.

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 prefetching the object data for that DSMCC object into cache. The returned value is true if the MHP supports prefetching (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 prefetching. 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.

Parameters:

`DSMCCObject` - the directory object in which to prefetch the data.

`path` - the relative path name of object to prefetch, starting from the directory object passes as parameter.

`priority` - the relative priority of this prefetch request (higher = more important)

---

prefetch(String, byte)

```
public static boolean prefetch(java.lang.String path, byte priority)
```

Calling this method will issue a hint to the MHP for prefetching the object data for that DSMCC object into cache. The returned value is true if the MHP supports prefetching (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 prefetching. 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.

Parameters:

`the` - absolute pathname of the object to prefetch. The path should be constructed as been defined for this API ('//VolumeName/pathName')

`priority` - the relative priority of this prefetch request (higher = more important)

---

removeObjectChangeEventListener(ObjectChangeEventListener)

```
public void removeObjectChangeEventListener(ObjectChangeEventListener listener)
```

Unsubscribes an ObjectChangeEventListener to receive notifications of version changes of DSMC-CObject.

Parameters:

`listener` - a previously registered ObjectChangeEventListener.

---

synchronousLoad()

```
public void synchronousLoad()
```

This fonction is used to load a DSMCCObject. It blocks until the file is loaded. It can be aborted from another thread with the abort method. In this case the InterruptedIOException is thrown. If the IOR of

the object itself or one of its parent directories is a Lite Option Profile Body, the MHP implementation will not attempt to resolve it : a ServiceXFRException is thrown to indicate to the application where the DSMCCObject is actually located.

Throws:

`CarouselException` - an error has occured during the loading.

NotEntitledException - the stream carrying the object is scramble and the user has no entitlements to descramble the stream

`InterruptedIOException` - the loading has been aborted.

InvalidPathNameException - the Object can not be found.

ServiceXFRException - the IOR of the object or one of its parent directories is a Lite Option Profile Body.

DSMCCException

---

unload()

```
public void unload()
```

When calling this method, the applications gives a hint to the MHP that if this object is not consumed by another application/thread, the system can free all the resources allocated to this object. It is worth noting that if other clients use this object (eg. a file input stream is opened on this object or if the corresponding stream or stream event is being consumed) the system resources allocated to this object will not be freed.

Throws:

NotLoadedException - the carousel object is not loaded.

org.dvb.dsmcc
DSMCCStream

Syntax
```
public class DSMCCStream

java.lang.Object
  |
  +--org.dvb.dsmcc.DSMCCStream
```

Direct Known Subclasses:
DSMCCStreamEvent

Description
The Stream class is used to manage DSMCC Stream Objects.

See Also:
DSMCCObject

Constructors

DSMCCStream(DSMCCObject)

```
public DSMCCStream(DSMCCObject aDSMCCObject)
```

Creates a Stream Object from a DSMCC Object. The DSMCCFile has to be a DSMCCStream (or a DSMCCStreamEvent)

Parameters:
`aDSMCCObject` - the DSMCC object which describes the stream

Throws:
NotLoadedException - the DSMCCObject is not loaded.

IllegalObjectTypeException - the DSMCCObject is neither a DSMCC Stream nor a DSMCCStreamEvent

---

DSMCCStream(String)

```
public DSMCCStream(java.lang.String path)
```

Create a Stream Object from its pathname. For an object Carousel, this method will block until the module which contains the object is loaded. The path has to lead to a DSMCCStream (or a DSMCCStreamEvent)

Parameters:
`path` - the pathname of the DSMCCStream Object.

Throws:
`IOException` - If an IO error occured.

IllegalObjectTypeException - the DSMCCObject is neither a DSMCC Stream nor a DSMCCStreamEvent

DSMCCStream(String, String)

```
public DSMCCStream(java.lang.String path, java.lang.String name)
```

Create a DSMCCStream from its pathname. For an object Carousel, this method will block until the module which contains the object is loaded. The path has to lead to a DSMCCStream (or a DSMCCStreamEvent)

Parameters:

`path` - the directory path.

`name` - the name of the DSMCCStream Object.

Throws:

`IOException` - If an IO error occured.

IllegalObjectTypeException - the DSMCCObject is neither a DSMCC Stream nor a DSMCCStreamEvent

Methods

getDescription()

```
public java.lang.String getDescription()
```

This function returns the description of the DSMCC stream object.

Returns:

The description of the DSMCC stream object.

getDuration()

```
public long getDuration()
```

This function returns the duration in milliseconds of the DSMCC Stream.

Returns:

The duration in milliseconds of the DSMCC Stream.

getNPT()

```
public long getNPT()
```

This function is used to get the current NPT in milliseconds.

Returns:

the current NPT in milliseconds.

Throws:

MPEGDeliveryException

getStreamLocator()

```
public org.davic.net.Locator getStreamLocator()
```

This function returned a Locator referencing the streams of this collection. The intepretation of the return value is determined by the isMPEGProgramStream() method.

Returns:

A locator.

isAudio()

```
public boolean isAudio()
```

This function returns a boolean indicating if the Stream Object refers to an audio stream

Returns:
true 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

Returns:
true if the Stream object refers to a data stream

isMPEGProgramStream()

```
public boolean isMPEGProgramStream()
```

This function returns a boolean indicating if this instance of DSMCCStream refers to a program stream (if the Stream Object contains a BIOP_PROGRAM_USE Tap) or a collection of elementary streams (if the Stream Object contains a list of BIOP_ES_USE Tap)

Returns:
true if the Stream object refers to a MPEG Program stream, false if it refers to one or more elementary streams

isVideo()

```
public boolean isVideo()
```

This function returns a boolean indicating if the Stream Object refers to an video stream

Returns:
true if the Stream object refers to an video stream

org.dvb.dsmcc
DSMCCStreamEvent

Syntax
```
public class DSMCCStreamEvent extends DSMCCStream

java.lang.Object
  |
  +--DSMCCStream
        |
        +--org.dvb.dsmcc.DSMCCStreamEvent
```

Description
The DSMCCStreamEvent class is used to manage DSMCC StreamEvent Objects.

Constructors

DSMCCStreamEvent(DSMCCObject)

```
public DSMCCStreamEvent(DSMCCObject aDSMCCObject)
```

Create a StreamEvent from a DSMCCObject. The Object has to be a DSMCC StreamEvent

Parameters:
`aDSMCCObject` - the DSMCC object which describes the stream.\

Throws:
NotLoadedException - the DSMCCObject is not loaded.

`IOException` - An IO error has occured.

IllegalObjectTypeException - the path does not lead to a DSMCC StreamEvent.

DSMCCStreamEvent(String)

```
public DSMCCStreamEvent(java.lang.String path)
```

Create a Stream Object from its pathname. For an object Carousel, this method will block until the module which contains the object is loaded. The path has to lead to a DSMCC Sttream Event

Parameters:
`path` - the pathname of the DSMCCStream Object.

Throws:
`IOException` - An IO error has occured.

IllegalObjectTypeException - the path does not lead to a DSMCC StreamEvent.

DSMCCStreamEvent(String, String)

```
public DSMCCStreamEvent(java.lang.String path, java.lang.String name)
```

Create a DSMCCStreamEvent from its pathname. For an object Carousel, this method will block until the module which contains the object is loaded. The path has to lead to a DSMCC Sttream Event

Parameters:
`path` - the directory path.

`name` - the name of the DSMCCStreamEvent Object.

Throws:
`IOException` - If an IO error occured.

IllegalObjectTypeException - the path does not lead to a DSMCC StreamEvent.

Methods

---

getEventList()

```
public java.lang.String[] getEventList()
```

This function is used to get the list of the events of the DSMCCStreamEvent object.

Returns:
The list of the eventName.

---

subscribe(String, StreamEventListener)

```
public synchronized int subscribe(java.lang.String eventName, StreamEventListener l)
```

This function is used to subscribe to an event of a DSMCC StreamEvent object.

Parameters:
`eventName` - the name of the event.

`l` - an object that implements the StreamEventListener Interface.

Returns:
The event Identifier.

Throws:
UnknownEventException - The event can not be found.

---

unsubscribe(int, StreamEventListener)

```
public synchronized void unsubscribe(int eventId, StreamEventListener l)
```

This function is used to cancel the subscription to an event of a DSMCCEvent object.

Parameters:
`eventId` - Identifier of the event.

`l` - an object that implements the StreamEventListener Interface.

Throws:
UnknownEventException - The event can not be found.

---

unsubscribe(String, StreamEventListener)

```
public synchronized void unsubscribe(java.lang.String eventName, StreamEventListener l)
```

This function is used to cancel the subscription to an event of a DSMCCEvent object.

Parameters:
`eventName` - the name of the event.

`l` - an object that implements the StreamEventListener Interface.

Throws:

UnknownEventException - The event can not be found.

org.dvb.dsmcc
IllegalObjectTypeException

Syntax

```
public class IllegalObjectTypeException extends DSMCCException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--DSMCCException
                          |
                          +--org.dvb.dsmcc.IllegalObjectTypeException
```

All Implemented Interfaces:

java.io.Serializable

Description

This Exception is thrown when the application attempted to create a DSMCCStream or DSM-CCStreamEvent 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
InvalidAddressException

Syntax
```
public class InvalidAddressException extends DSMCCException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--DSMCCException
                          |
                          +--org.dvb.dsmcc.InvalidAddressException
```

All Implemented Interfaces:
java.io.Serializable

Description
A InvalidAddressException is thrown when the format of an NSAP address is not recognized.

Constructors

InvalidAddressException()

```
public InvalidAddressException()
```

Construct a InvalidAddressException with no detail message

InvalidAddressException(String)

```
public InvalidAddressException(java.lang.String s)
```

Construct a InvalidAddressException with the specified detail message

Parameters:
`s` - the detail message

org.dvb.dsmcc
InvalidFormatEvent

Syntax
```
public class InvalidFormatEvent extends AsynchronousLoadingEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--AsynchronousLoadingEvent
              |
              +--org.dvb.dsmcc.InvalidFormatEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
The format of the data received is inconsistent

Constructors

InvalidFormatEvent(DSMCCObject)

```
public InvalidFormatEvent(DSMCCObject o)
```

The constructor for this event

Methods

---

getSource()

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

Returns the DSMCCObject that generated the event

Overrides:
getSource() in class AsynchronousLoadingEvent

org.dvb.dsmcc
InvalidFormatException

Syntax
```
public class InvalidFormatException extends DSMCCException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--DSMCCException
                          |
                          +--org.dvb.dsmcc.InvalidFormatException
```

All Implemented Interfaces:
java.io.Serializable

Description
An InvalidFormatException is thrown when an inconsistent DSMCC message is received.

Constructors

InvalidFormatException()

```
public InvalidFormatException()
```

Construct an InvalidFormatException with no detail message

InvalidFormatException(String)

```
public InvalidFormatException(java.lang.String s)
```

Construct an InvalidFormatException with the specified detail message

Parameters:
`s` - the detail message

org.dvb.dsmcc
InvalidPathnameEvent

Syntax
```
public class InvalidPathnameEvent extends AsynchronousLoadingEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--AsynchronousLoadingEvent
              |
              +--org.dvb.dsmcc.InvalidPathnameEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
The pathname doe not exist or the ServiceDomain has been detached.

Constructors

InvalidPathnameEvent(DSMCCObject)

```
public InvalidPathnameEvent(DSMCCObject o)
```

The constructor for this event

Methods

---

getSource()

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

Returns the DSMCCObject that generated the event

Overrides:
getSource() in class AsynchronousLoadingEvent

org.dvb.dsmcc
InvalidPathNameException

Syntax
```
public class InvalidPathNameException extends DSMCCException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--DSMCCException
                          |
                          +--org.dvb.dsmcc.InvalidPathNameException
```

All Implemented Interfaces:
java.io.Serializable

Description
The InvalidPathNameException is thrown when the PathName 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
MPEGDeliveryErrorEvent

Syntax
```
public class MPEGDeliveryErrorEvent extends AsynchronousLoadingEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--AsynchronousLoadingEvent
              |
              +--org.dvb.dsmcc.MPEGDeliveryErrorEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
Error (for instance, a time out or accessing the data would require tuning) has occured while loading data from an MPEG Stream.

Constructors

MPEGDeliveryErrorEvent(DSMCCObject)

```
public MPEGDeliveryErrorEvent(DSMCCObject o)
```

The constructor for this event

Methods

---

getSource()

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

Returns the DSMCCObject that generated the event

Overrides:
getSource() in class AsynchronousLoadingEvent

org.dvb.dsmcc
MPEGDeliveryException

Syntax
```
public class MPEGDeliveryException extends DSMCCException

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--DSMCCException
                          |
                          +--org.dvb.dsmcc.MPEGDeliveryException
```

All Implemented Interfaces:
java.io.Serializable

Description
An MPEGDEliveryException is thrown when an error (for instance, a time out or accessing the data would require tuning) occurs while loading data from an MPEG Stream.

Constructors

MPEGDeliveryException()

```
public MPEGDeliveryException()
```

Construct an MPEGDeliveryException with no detail message

MPEGDeliveryException(String)

```
public MPEGDeliveryException(java.lang.String s)
```

Construct an MPEGDeliveryException with the specified detail message

Parameters:
`s` - the detail message

org.dvb.dsmcc
NotEntitledEvent

Syntax
```
public class NotEntitledEvent extends AsynchronousLoadingEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--AsynchronousLoadingEvent
              |
              +--org.dvb.dsmcc.NotEntitledEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
This event is sent when an attempt to asynchronously load an object has failed because the elementary STream carrying the object is scrambled and the user is not entitled to access the content of the object.

Constructors

NotEntitledEvent(DSMCCObject)

```
public NotEntitledEvent(DSMCCObject o)
```

The constructor for this event

Methods

---

getSource()

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

Returns the DSMCCObject that generated the event

Overrides:
getSource() in class AsynchronousLoadingEvent

org.dvb.dsmcc
NotEntitledException

Syntax

```
public class NotEntitledException extends DSMCCException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--DSMCCException
                          |
                          +--org.dvb.dsmcc.NotEntitledException
```

All Implemented Interfaces:

java.io.Serializable

Description

A This Exception is thrown when the user is not entitled to access the content of the object (the Elementary Stream is scrambled and the user is not entitled)

Constructors

NotEntitledException()

```
public NotEntitledException()
```

construct a NotEntitledException with no detail message

NotEntitledException(String)

```
public NotEntitledException(java.lang.String s)
```

construct a NotEntitledException with a detail message

Parameters:

`s` - detail message

org.dvb.dsmcc
NothingToAbortException

Syntax
```
public class NothingToAbortException extends DSMCCException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--DSMCCException
                          |
                          +--org.dvb.dsmcc.NothingToAbortException
```

All Implemented Interfaces:
java.io.Serializable

Description
A NothingToAbortException is thrown when the abort method is called and there is no loading in progress.

Constructors

NothingToAbortException()

```
public NothingToAbortException()
```

Construct a NothingToAbortException with no detail message

NothingToAbortException(String)

```
public NothingToAbortException(java.lang.String s)
```

Construct a NothingToAbortException with the specified detail message

Parameters:
`s` - the detail message

org.dvb.dsmcc
NotLoadedException

Syntax
```
public class NotLoadedException extends DSMCCException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--DSMCCException
                          |
                          +--org.dvb.dsmcc.NotLoadedException
```

All Implemented Interfaces:
java.io.Serializable

Description
A NotLoadedException is thrown when the Stream object constructor is called with a DSMCC Object which is not loaded.

Constructors

NotLoadedException()

```
public NotLoadedException()
```

Construct a NotLoadedException with no detail message

NotLoadedException(String)

```
public NotLoadedException(java.lang.String s)
```

Construct a NotLoadedException with the specified detail message

Parameters:
`s` - the detail message

org.dvb.dsmcc
ObjectChangeEvent

Syntax
```
public class ObjectChangeEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.dsmcc.ObjectChangeEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
This class describes an ObjectChange event that is used to monitor the arrival of a new version of a DSMCC Object in an Object Carousel.

Constructors

ObjectChangeEvent(DSMCCObject, int)

```
public ObjectChangeEvent(DSMCCObject source, int aVersionNumber)
```

Creates an ObjectChangeEvent indicating that a new version of the monitored DSMCC Object has been detected. It is up to the application to reload the new version of the object.

Parameters:

`source` - the DSMCCObject whose version has changed

`aVersionNumber` - the new version number.

Methods

---

getNewVersionNumber()

```
public int getNewVersionNumber()
```

This method is used to get the new version number of the monitored DSMCC Object.

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

org.dvb.dsmcc
ObjectChangeEventListener

Syntax
```
public interface ObjectChangeEventListener extends java.util.EventListener
```

All Superinterfaces:
java.util.EventListener

Description
The objects that implements the ObjectChangeEventListener interface can receive ObjectChangeEvent event.

Methods

---

receiveObjectChangeEvent(ObjectChangeEvent)

```
public void receiveObjectChangeEvent(ObjectChangeEvent e)
```

Send a ObjectChangeEvent to the ObjectChangeEventListener.

Parameters:
`e` - the ObjectChangeEvent event.

org.dvb.dsmcc
ServerDeliveryErrorEvent

Syntax
```
public class ServerDeliveryErrorEvent extends AsynchronousLoadingEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--AsynchronousLoadingEvent
              |
              +--org.dvb.dsmcc.ServerDeliveryErrorEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
The local machine can not communicate with the server. This event is only used with IIOP, for the object carousel the MPEGDeliveryErrorEvent is used instead.

Constructors

ServerDeliveryErrorEvent(DSMCCObject)

```
public ServerDeliveryErrorEvent(DSMCCObject o)
```

The constructor for this event

Methods

---

getSource()

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

Returns the DSMCCObject that generated the event

Overrides:
getSource() in class AsynchronousLoadingEvent

org.dvb.dsmcc
ServerDeliveryException

Syntax
```
public class ServerDeliveryException extends DSMCCException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--DSMCCException
                          |
                          +--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 IIOP, for the object carousel the MPEGDeliveryException is used instead.

Constructors

ServerDeliveryException()

```
public ServerDeliveryException()
```

Construct a ServerDeliveryException with no detail message

ServerDeliveryException(String)

```
public ServerDeliveryException(java.lang.String s)
```

Construct a ServerDeliveryException with the specified detail message

Parameters:
`s` - the detail message

org.dvb.dsmcc
ServiceDomain

Syntax
```
public class ServiceDomain

java.lang.Object
  |
  +--org.dvb.dsmcc.ServiceDomain
```

Description
A ServiceDomain is a group of DSMCC object. The objects are sent either with the Object Carousel for a broadcast Network or with the IIOP protocol for an interactive network. *To access the objects of a ServiceDomain, it has to be attached to the Filesystem hierarchy. As soon as it is attached, the objects can be reached through the absolute path //VolumeName/aPathName with the java.io package. By definition, the use of the VolumeName "//Home/" enables an application to address objects in the same carousel as the one that contains it. * To access the content of an object, the application has four ways: - It can instanciate the class that is used to read the object (java.io.FileInputStream or java.io.RandomAccessFile for a File or DSMCCStream for aStream) 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 instanciate the class used to read the object. - It can instanciate 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.

Constructors

ServiceDomain(String)

```
public ServiceDomain(java.lang.String VolumeName)
```

Parameters:

`VolumeName` - the Volume name that will be mounted when the ServiceDomain is attached to the filesystem hierarchy.

Methods

---

attach(byte[])

```
public void attach(byte[] NSAPAddress)
```

This function is used to attach a ServiceDomain from either an objectCarousel or from an interactive network. All the objects of the serviceDomain can be reached with the path //VolumeName/. This call will block until the attachement is done.

Parameters:

`NSAPAddress` - The NSAP Address of a ServiceDomain as defined in in ISO/IEC 13818-6

Throws:

`InterruptedIOException` - The attachement has been aborted.

InvalidAddressException - The NSAP Address is invalid.

DSMCCException - An error has occured during the attachement.

MPEGDeliveryException - attaching to this domain would require tuning.

---

attach(Locator)

```
public void attach(org.davic.net.Locator l)
```

This function is used to get a ServiceDomain from an object Carousel. It loads the module which contains the Service Gateway object and mounts the ServiceDomain Volume in the filesystem hierarchy. All the objects of the serviceDomain can be reached with the path //VolumeName/. This call will block until the ServiceGateway is loaded. It can be aborted by another thread with the method detach. In this case an InterruptedIOException is thrown.

Parameters:

`l` - The locator pointing to the elementary stream carrying the DSI of the Object Carousel, or to a DVBService that carries one and only one Object Carousel.

Throws:

DSMCCException - An error has occurred during the attachement (eg the locator does not point to a component carrying a DSI of an Object Carousel or a service containing a carousel)

`InterruptedIOException` - The attachment has been aborted.

MPEGDeliveryException - attaching to this domain would require tuning.

---

attach(Locator, int)

```
public void attach(org.davic.net.Locator aDVBService, int aCarouselId)
```

This function is used to get a ServiceDomain from an object Carousel. It loads the module which contains the Service Gateway object and mounts the ServiceDomain Volume in the filesystem hierarchy. All the objects of the serviceDomain can be reached with the path //VolumeName/. This call will block until the ServiceGateway is loaded. It can be aborted by another thread with the method detach. In this case an InterruptedIOException is thrown.

Parameters:

`aDVBService` - The coordinates of the DVBService which contains the Object Carousel. This locator has to point to a DVBService.

`aCarouselId` - The identifier of the carousel.

Throws:

DSMCCException - An error has occured during the attachement.

`InterruptedIOException` - The attachment has been aborted.

MPEGDeliveryException - attaching to this domain would require tuning.

---

detach()

```
public void detach()
```

A call to this method is a hint that the applications gives to the MHP to unmount the volume and delete the objects of the service domain. When another application is using objects of the same service domain the method has no effects. When there are no other application using objects of the service domain, a call to this method is a hint that the MHP can free all the resources allocated to this sevice domain."

Throws:

NotLoadedException - is thrown if the ServiceDomain is not attached or if there is no attached in progress.

---

getNSAPAddress()

```
public byte[] getNSAPAddress()
```

This method return the NSAP address of the ServiceDomain.

Throws:

NotLoadedException - is thrown if the ServiceDomain is not attached.

org.dvb.dsmcc
ServiceXFRErrorEvent

Syntax

```
public class ServiceXFRErrorEvent extends AsynchronousLoadingEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--AsynchronousLoadingEvent
              |
              +--org.dvb.dsmcc.ServiceXFRErrorEvent
```

All Implemented Interfaces:

java.io.Serializable

Description

The object requested is available in an alternate ServiceDomain. When an application attempts to asynchronously load an object that has itself a LiteOptionProfileBody IOR or that has a parent directory that has a LiteOptionProfileBody IOR, this event shall be sent to the application. There is no implicit mounting by the implementation of the carousel that actually contain the object. This event is also sent even if the Service Domain that actually contain the DSMCCObject is already mounted.

Constructors

ServiceXFRErrorEvent(DSMCCObject, ServiceXFRReference)

```
public ServiceXFRErrorEvent(DSMCCObject o, ServiceXFRReference ref)
```

The constructor for this event

Methods

getServiceXFR()

```
public ServiceXFRReference getServiceXFR()
```

This method is used to get a reference to the service domain that contains the requested onject.

Returns:

The adress 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

org.dvb.dsmcc
ServiceXFRException

Syntax
```
public class ServiceXFRException extends DSMCCException

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--DSMCCException
                          |
                          +--org.dvb.dsmcc.ServiceXFRException
```

All Implemented Interfaces:
java.io.Serializable

Description
A ServiceXFRException is thrown when a DSMCC Object can not be loaded in the current ServiceDomain but is available in an alternate ServiceDomain (i.e. For an object Carousel, the IOR of the object or one of its parent directories contains a Lite Option Profile Body). There is no implicit mounting by the implementation of the carousel that actually contain the object. This exception is also thrown even if the Service Domain that actually contain the dSMCCObject is already mounted.

Constructors

ServiceXFRException(byte[], String)

```
public ServiceXFRException(byte[] NSAPAddress, java.lang.String pathName)
```

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)
```

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. return the adress of an alternate ServiceDomain where the object can be found.

org.dvb.dsmcc
ServiceXFRReference

Syntax
```
public class ServiceXFRReference

java.lang.Object
  |
  +--org.dvb.dsmcc.ServiceXFRReference
```

Description
A ServiceXFRReference object is used when a DSMCC Object can not be loaded in the current Service-Domain but is available in an alternate ServiceDomain.

Constructors

ServiceXFRReference(byte[], String)

```
public ServiceXFRReference(byte[] NSAPAddress, java.lang.String pathName)
```

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 aService, int aCarouselId,
            java.lang.String aPathName)
```

Parameters:

`aService` - Locator of the Service

`carouselId` - Carousel Identifier

`pathName` - pathName of the object in the alternate ServiceDomain

Methods

---

getCarouselId()

```
public int getCarouselId()
```

Returns:

The carousel identifier

---

getLocator()

```
public org.davic.net.Locator getLocator()
```

Returns:

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()
```

Returns:
The NSAP Address of a ServiceDomain as defined in ISO/IEC 13818-6

---

getPathName()

```
public java.lang.String getPathName()
```

Returns:
The pathName of the object in the alternate ServiceDomain

org.dvb.dsmcc
StreamEvent

Syntax
```
public class StreamEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.dsmcc.StreamEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
This class describes a Stream event which is used to synchronize an application with an MPEG Stream.

Constructors

StreamEvent(DSMCCStreamEvent, long, String, int, byte[])

```
public StreamEvent(DSMCCStreamEvent source, long NPT, java.lang.String name, int eventId,
            byte[] eventData)
```

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 Event

Returns:
The Identifier of the Event

---

getEventName()

```
public java.lang.String getEventName()
```

This method is used to get the name of the Event

Returns:
the name of the Event

---

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()
```

Returns the DSMCCStreamEvent that generated the event

Overrides:
java.util.EventObject.getSource() in class java.util.EventObject

org.dvb.dsmcc
StreamEventListener

Syntax
```
public interface StreamEventListener extends java.util.EventListener
```

All Superinterfaces:
java.util.EventListener

Description
The objects that implements the StreamEventListener interface can receive StreamEvent event.

Methods

---

receiveStreamEvent(StreamEvent)

```
public void receiveStreamEvent(StreamEvent e)
```

Send a StreamEvent to the StreamEventListener.

Parameters:
`e` - the StreamEvent event.

org.dvb.dsmcc
SuccessEvent

Syntax
```
public class SuccessEvent extends AsynchronousLoadingEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--AsynchronousLoadingEvent
              |
              +--org.dvb.dsmcc.SuccessEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
This event indicates that the asynchronous loading was successful.

Constructors

SuccessEvent(DSMCCObject)

```
public SuccessEvent(DSMCCObject o)
```

The constructor for this event

Methods

---

getSource()

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

Returns the DSMCCObject which was successfully loaded.

Overrides:
getSource() in class AsynchronousLoadingEvent

Returns:
the loaded CarouselObject.

org.dvb.dsmcc
UnknownEventException

Syntax
```
public class UnknownEventException extends DSMCCException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--DSMCCException
                          |
                          +--org.dvb.dsmcc.UnknownEventException
```

All Implemented Interfaces:
java.io.Serializable

Description
The UnknownEventException is thrown when a method tries to access to an unknown event.

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

Class Summary
Classes
DatagramSocketBuffer-Control

org.dvb.net
DatagramSocketBufferControl

Syntax

```
public class DatagramSocketBufferControl

java.lang.Object
  |
  +--org.dvb.net.DatagramSocketBufferControl
```

Methods

---

getReceiveBufferSize(DatagramSocket)

```
public static int getReceiveBufferSize(java.net.DatagramSocket d)
```

Get value of the SO_RCVBUF option for this socket, that is the buffer size used by the platform for input on the this Socket.

Parameters:

`d` - The DatagramSocket for which to query the receive buffer size.

Returns:

The size of the receive buffer, in bytes.

Throws:

`SocketException` - - If there is an error when querying the SO_RCVBUF option.

---

setReceiveBufferSize(DatagramSocket, int)

```
public static void setReceiveBufferSize(java.net.DatagramSocket d, int size)
```

Sets the SO_RCVBUF option to the specified value for this DatagramSocket. The SO_RCVBUF option is used by the platform's networking code as a hint for the size to use to allocate set 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 maximum size of a packet that may be sent on this socket.

Because SO_RCVBUF is a hint, applications that want to verify what size the buffers were set to should call getReceiveBufferSize.

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.

`IllegalArgumentException` - - If size is 0 or is negative.

Annex R (normative): DVB-J Return Channel Connection Management API

Package org.dvb.net.rc

Class Summary

Interfaces

ConnectionListener	This interface should be implemented by objects wishing to be notified about the connection status of a ConnectionRCInterface.

Classes

ConnectionEstablishedEvent	ConnectionEstablishedEvent - An event generated after a connection is established for a ConnectionRCInterface.
ConnectionFailedEvent	ConnectionFailedEvent - An event generated after an attempt to setup a connection for a ConnectionRCInterface fails.
ConnectionParameters	This class encapsulates the parameters needed to specify the target of a connection.
ConnectionRCEvent	ConnectionRCEvent - the base class for events related to connection oriented return channels.
ConnectionRCInterface	This class models a connection based return channel network interface for use in receiving and transmitting IP packets over a return channel.
ConnectionTerminatedEvent	ConnectionTerminatedEvent - An event generated after a connected ConnectionRCInterface is disconnected.
RCInterface	This class models a return channel network interface for use in receiving and transmitting IP packets over a logical return channel.
RCInterfaceManager	This class is the factory and manager for all return channel interfaces in the system.
RCInterfaceReleasedEvent	This event informs an application that a RCInterface has been released by an application or other entity in the system.
RCInterfaceReservedEvent	This event informs an application that a RCInterface has been reserved by an application or other entity in the system.
RCPermission	This class is for return channel set-up permissions.

Exceptions

IncompleteTargetException	Thrown when the target for a connection is incompletely specified.
PermissionDeniedException	Thrown when an application calls a method which it does not have permission to call at that time.

org.dvb.net.rc
ConnectionEstablishedEvent

Syntax
```
public class ConnectionEstablishedEvent extends ConnectionRCEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--ConnectionRCEvent
              |
              +--org.dvb.net.rc.ConnectionEstablishedEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
ConnectionEstablishedEvent - An event generated after a connection is established for a `ConnectionRCInterface`.

Constructors

ConnectionEstablishedEvent(Object)

```
public ConnectionEstablishedEvent(java.lang.Object source)
```

Construct an event.

Parameters:
`source` - the `ConnectionRCInterface` whose connection was established

org.dvb.net.rc
ConnectionFailedEvent

Syntax

```
public class ConnectionFailedEvent extends ConnectionRCEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--ConnectionRCEvent
              |
              +--org.dvb.net.rc.ConnectionFailedEvent
```

All Implemented Interfaces:

java.io.Serializable

Description

ConnectionFailedEvent - An event generated after an attempt to setup a connection for a `ConnectionRCInterface` fails.

Constructors

ConnectionFailedEvent(Object)

```
public ConnectionFailedEvent(java.lang.Object source)
```

Construct an event.

Parameters:

`source` - the `ConnectionRCInterface` whose connection attempt failed

org.dvb.net.rc
ConnectionListener

Syntax
```
public interface ConnectionListener
```

Description
This interface should be implemented by objects wishing to be notified about the connection status of a `ConnectionRCInterface`.

Methods

---

connectionChanged(ConnectionRCEvent)

```
public void connectionChanged(ConnectionRCEvent e)
```

This method is called to report events related to the setup and termination of return channel interface connections.

org.dvb.net.rc
ConnectionParameters

Syntax

```
public class ConnectionParameters

java.lang.Object
  |
  +--org.dvb.net.rc.ConnectionParameters
```

Description

This class encapsulates the parameters needed to specify the target of a connection.

Constructors

ConnectionParameters(String, String, String, 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 address to use for the DNS server

Methods

---

getDNSServer()

```
public java.net.InetAddress getDNSServer()
```

Return the address of the DNS server to use for the connection

Parameters:

`return` - the address of the DNS server to use for the connection

---

getPassword()

```
public java.lang.String getPassword()
```

Return the password used in establishing this connection

Parameters:

`return` - the password used in establishing this connection

---

getTarget()

```
public java.lang.String getTarget()
```

Return the target of this connection for example a phone number

Parameters:

`return` - the target of this connection

---

getUsername()

```
public java.lang.String getUsername()
```

Return the username used in establishing this connection

Parameters:

`return` - the username used in establishing this connection

org.dvb.net.rc
ConnectionRCEvent

Syntax
```
public class ConnectionRCEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.net.rc.ConnectionRCEvent
```

Direct Known Subclasses:
ConnectionEstablishedEvent, ConnectionFailedEvent, ConnectionTerminatedEvent

All Implemented Interfaces:
java.io.Serializable

Description
ConnectionRCEvent - the base class for events related to connection oriented return channels.

Constructors

ConnectionRCEvent(Object)

```
public ConnectionRCEvent(java.lang.Object source)
```

Construct an event

Parameters:

`source` - the `ConnectionRCInterface` for which the event was generated.

org.dvb.net.rc
ConnectionRCInterface

Syntax
```
public class ConnectionRCInterface extends RCInterface implements
    org.davic.resources.ResourceProxy

java.lang.Object
  |
  +--RCInterface
        |
        +--org.dvb.net.rc.ConnectionRCInterface
```

All Implemented Interfaces:
org.davic.resources.ResourceProxy

Description
This class models a connection based return channel network interface for use in receiving and transmitting IP packets over a return channel. Targets for connections are specified as strings including the number to dial. These strings can only include numbers and are passed direct to the underlying device.

Methods

---

addConnectionListener(ConnectionListener)

```
public void addConnectionListener(ConnectionListener l)
```

Add a listener for events related to connections of this interface.

Parameters:
`l` - the listener for the connection related events

---

connect()

```
public void connect()
```

Connect this return channel to the current target. If at the time the method is called, it is known that connection is impossible then an exception will be thrown. If this ResourceProxy does not have the underlying resource reserved then a PermissionDeniedException will be thrown. Apart from this, this method is asynchronous and completion or failure is reported through the event listener on this class.

Throws:
`IOException` - if connection is known to be impossible at the time when the method is called

PermissionDeniedException - if this application does not own the resource

`IOException`

---

disconnect()

```
public void disconnect()
```

Disconnect this return channel from the current target. This method is asynchronous and completion is reported through the event listener on this class. This method does not release the underlying resource from the ResourceProxy.

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 withdrawl of the underlying resource. This is the object provided as the parameter to last call to the reserve method on this object. If this object is not currently bound to the underlying resource then null is returned.

Specified By:
org.davic.resources.ResourceProxy.getClient() in interface org.davic.resources.ResourceProxy

Returns:
the object which asked to be notified about withdrawal of the underlying physical resource from this resource proxy or null

---

getConnectedTime()

```
public int getConnectedTime()
```

Return the time an interface has been connected

Returns:
the time in seconds for which this interface has been connected or -1 if the device is not connected

---

getCurrentTarget()

```
public ConnectionParameters getCurrentTarget()
```

Get the current target for connections. This method may throw a SecurityException if the application is not allowed to read the current target.

Returns:
the current set of connection target parameters

Throws:
IncompleteTargetException - if the current target is not completely configured

---

getSetupTimeEstimate()

```
public float getSetupTimeEstimate()
```

Obtain an estimate of the connection time for this interface in seconds.

Returns:
an estimate of the connection time 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 is ignored.

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 synchronized void reserve(org.davic.resources.ResourceClient c,
            java.lang.Object requestData)
```

Request the right to control this return channel interface. This method may throw a SecurityException if the application is denied access to the resource by security policy.

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

setTarget(ConnectionParameters)

```
public void setTarget(ConnectionParameters target)
```

Set a non-default target for connections. This method may throw a SecurityException if the application is not allowed to modify the target.

Parameters:

`target` - the new set of connection target parameters

Throws:

IncompleteTargetException - if the target is not completely specified

PermissionDeniedException - if this application does not own the resource

setTargetToDefault()

```
public void setTargetToDefault()
```

Set the target for connections to the default. This method may throw a SecurityException if the application is not allowed to connect to the default.

Throws:

IncompleteTargetException - if the target is not completely specified

PermissionDeniedException - if this application does not own the resource

org.dvb.net.rc
ConnectionTerminatedEvent

Syntax

```
public class ConnectionTerminatedEvent extends ConnectionRCEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--ConnectionRCEvent
              |
              +--org.dvb.net.rc.ConnectionTerminatedEvent
```

All Implemented Interfaces:

java.io.Serializable

Description

ConnectionTerminatedEvent - An event generated after a connected `ConnectionRCInterface` is disconnected.

Constructors

ConnectionTerminatedEvent(Object)

```
public ConnectionTerminatedEvent(java.lang.Object source)
```

Construct an event.

Parameters:

`source` - the `ConnectionRCInterface` whose status changed

org.dvb.net.rc
IncompleteTargetException

Syntax

```
public class IncompleteTargetException extends java.lang.Exception

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.dvb.net.rc.IncompleteTargetException
```

All Implemented Interfaces:

java.io.Serializable

Description

Thrown when the target for a connection is incompletely specified. This is thrown either when the default connection target is incompletely defined in the device or when an application provides an incompletely defined connection target to the device.

Constructors

IncompleteTargetException()

```
public IncompleteTargetException()
```

Default constructor for the exception

IncompleteTargetException(String)

```
public IncompleteTargetException(java.lang.String reason)
```

Constructor for the exception with a specified reason

Parameters:

`reason` - the reason why the exception was raised

org.dvb.net.rc
PermissionDeniedException

Syntax
```
public class PermissionDeniedException extends java.lang.Exception

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.dvb.net.rc.PermissionDeniedException
```

All Implemented Interfaces:
java.io.Serializable

Description
Thrown when an application calls a method which it does not have permission to call at that time.

Constructors

PermissionDeniedException()

```
public PermissionDeniedException()
```

Default constructor for the exception

PermissionDeniedException(String)

```
public PermissionDeniedException(java.lang.String reason)
```

Constructor for the exception with a specified reason

Parameters:
`reason` - the reason why the exception was raised

org.dvb.net.rc
RCInterface

Syntax
```
public class RCInterface

java.lang.Object
  |
  +--org.dvb.net.rc.RCInterface
```

Direct Known Subclasses:
ConnectionRCInterface

Description
This class models a return channel network interface for use in receiving and transmitting IP packets over a logical return channel. This can include real analog modems, cable return channel and all the other options allowed by the relevant DVB specification. This class does not model any concept of connection. Hence interfaces represented by this class and not by a sub-class of it are permanently connected.

Fields

TYPE_CATV
```
public static final int TYPE_CATV
```
Constant to indicate a CATV return channel.

TYPE_DECT
```
public static final int TYPE_DECT
```
Constant to indicate a DECT return channel.

TYPE_ISDN
```
public static final int TYPE_ISDN
```
Constant to indicate an ISDN return channel.

TYPE_LMDS
```
public static final int TYPE_LMDS
```
Constant to indicate a LMDS return channel.

TYPE_MATV
```
public static final int TYPE_MATV
```
Constant to indicate a MATV return channel.

TYPE_PSTN

```
public static final int TYPE_PSTN
```

Constant to indicate a PSTN return channel.

Methods

getType()

```
public int getType()
```

Return the type of return channel represented by this object.

Returns:

the type of return channel represented by this object encoded as one of the constants defined in this class

org.dvb.net.rc
RCInterfaceManager

Syntax
```
public class RCInterfaceManager implements org.davic.resources.ResourceServer

java.lang.Object
  |
  +--org.dvb.net.rc.RCInterfaceManager
```

All Implemented Interfaces:
org.davic.resources.ResourceServer

Description
This class is the factory and manager for all return channel interfaces in the system. The methods on this class which return instances of the `RCInterface` will only return new instances of that class under the following conditions :-

- on the first occasion an instance needs to be returned to a particular application for a particular interface.
- when new return channel interfaces are added to the system

Methods

---

addResourceStatusEventListener(ResourceStatusListener)

```
public void addResourceStatusEventListener(org.davic.resources.ResourceStatusListener
            listener)
```

This method informs a resource server that a particular object should be informed of changes in the state of the resources managed by that server.

Specified By:
org.davic.resources.ResourceServer.addResourceStatusEventListener(org.davic.resources.ResourceStatusListener) in interface org.davic.resources.ResourceServer

Parameters:
`listener` - the object to be informed of state changes

---

getInstance()

```
public static RCInterfaceManager getInstance()
```

Factory method to obtain a manager

---

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

Throws:
`NoRouteToHostException` - if there is no route to the host concerned

---

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 it's interest initially then this method has no effect.

Specified By:
org.davic.resources.ResourceServer.removeResourceStatusEventListener(org.davic.resources.ResourceStatusListener) in interface org.davic.resources.ResourceServer

Parameters:
`listener` - the object which is no longer interested

org.dvb.net.rc
RCInterfaceReleasedEvent

Syntax

```
public class RCInterfaceReleasedEvent extends org.davic.resources.ResourceStatusEvent

java.lang.Object
  |
  +--org.davic.resources.ResourceStatusEvent
        |
        +--org.dvb.net.rc.RCInterfaceReleasedEvent
```

Description

This event informs an application that a `RCInterface` has been released by an application or other entity in the system. It is generated when an application which had successfully reserved a `RCInterface` calls the `ConnectionRCInterface.release` method. It will also be generated if any other entities in the system own such an interface and then release that interface in such a way that it could then become available to applications using this API.

Constructors

RCInterfaceReleasedEvent(Object)

```
public RCInterfaceReleasedEvent(java.lang.Object bg)
```

Constructor for the event

Parameters:

`bg` - the `RCInterface` which has been released

Methods

---

getSource()

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

Returns the device that has been released

Overrides:

org.davic.resources.ResourceStatusEvent.getSource() in class org.davic.resources.ResourceStatusEvent

Returns:

the `RCInterface` object representing the interface that has been released

org.dvb.net.rc
RCInterfaceReservedEvent

Syntax

```
public class RCInterfaceReservedEvent extends org.davic.resources.ResourceStatusEvent

java.lang.Object
  |
  +--org.davic.resources.ResourceStatusEvent
        |
        +--org.dvb.net.rc.RCInterfaceReservedEvent
```

Description

This event informs an application that a `RCInterface` has been reserved by an application or other entity in the system. It is generated when an application successfully reserves a `RCInterface`. It will also be generated if any other entities in the system reserve such an interface with the effect of something which was visible to applications using this API becoming unavailable.

Constructors

RCInterfaceReservedEvent(Object)

```
public RCInterfaceReservedEvent(java.lang.Object bg)
```

Constructor for the event

Parameters:

`bg` - the `RCInterface` representing the device which has been reserved

Methods

---

getSource()

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

Returns the device that has been reserved

Overrides:

org.davic.resources.ResourceStatusEvent.getSource() in class org.davic.resources.ResourceStatusEvent

Returns:

an `RCInterface` representing the device that has been reserved

org.dvb.net.rc
RCPermission

Syntax
```
public class RCPermission extends java.security.BasicPermission

java.lang.Object
  |
  +--java.security.Permission
        |
        +--java.security.BasicPermission
              |
              +--org.dvb.net.rc.RCPermission
```

All Implemented Interfaces:
java.security.Guard, java.io.Serializable

Description
This class is for return channel set-up permissions. An RCPermission contains a name, but no actions list.

The permission name can be "target:default", which indicates the permission to use the default connection parameters.

The permission name can also be "target:<phone number>", which indicates the permission to use the specified phone number in the connection set-up (ConnectionRCInterface.setTarget(ConnectionParameters) method).

A wildcard may be used at the end of the permission name. In that case, all phone numbers starting with the number before the wildcard are included in the permission.

Examples:

- target:0206234342 (Permission to dial the specified phone number)
- target:020* (Permission to dial phone numbers starting with 020)
- target:* (Permission to dial all phone numbers, including the default)

Note: ConnectionRCInterface.reserve(ResourceClient, Object) will throw a SecurityException if the application is not allowed to set-up a connection over the return channel at all (i.e., there is no valid target allowed).

Constructors

RCPermission(String)
```
public RCPermission(java.lang.String name)
```
Creates a new RCPermission with the specified name. The name is the symbolic name of the RCPermission.

Parameters:
`name` - the name of the RCPermission

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.

Annex S (normative): Application Listing and Launching

Package org.dvb.application

Class Summary

Interfaces

AppAttributes	The AppAttributes class is a mapping of various information about a registered application.
AppProxy	An AppProxy Object is a proxy to an application.
AppsDatabaseEventListener	The AppsDatabaseListener class allows an application to monitor the application database so that it can keep an up to date interface without polling the state.
AppStateChangeEventListener	The AppStateChangeEventListener class allows a launcher application to keep track of applications it launches.
DVBHTMLProxy	
DVBJProxy	

Classes

AppIcon	The AppIcon encapsulates the information concerning the icon attached to the application
AppID	The AppID is a representation of the unique identifier for applications.
AppsControlPermission	This class represents a Permission to control the lifecyle of another application
AppsDatabase	The AppsDatabase is an abstract view of the currently available applications.
AppsDatabaseEvent	The AppsDatabaseEvent class indicates either the an entry in the application database has changed, or so many changes have occurred.
AppsDatabaseFilter	Abstract class for the filters.
AppStateChangeEvent	The AppStateChangeEvent class indicates a state transition of the application.
CurrentServiceFilter	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)

Exceptions

IllegalProfileParameterException	The IllegalProfileParameter exception is thrown if the application attempts to ask for a version number for a profile not specified for the application.
LanguageNotAvailableException	The LanguageNotAvailableException exception is thrown if the application asks for the name of an application in a language not signalled in the AIT.
ProxyInUseException	The ProxyInUseException exception is thrown if the application attempts to alter the registration for an application but another application is currently holding a proxy to the same application, or the application is running

org.dvb.application
AppAttributes

Syntax
```
public interface AppAttributes
```

Description
The `AppAttributes` class is a mapping of various information about a registered application.

Since:
MHP1.0

Fields

DVB_HTML_application
```
public static final int DVB_HTML_application
```

DVB_J_application
```
public static final int DVB_J_application
```

Methods

getAppIcon()
```
public AppIcon getAppIcon()
```
This method returns the information related to the icons that are attached to the application.

Since:
MHP1.0

getIdentifier()
```
public AppID getIdentifier()
```
This method returns the application identifier depending on the application type.

Since:
MHP1.0

getIsServiceBound()
```
public boolean getIsServiceBound()
```
This method determines whether the application is signalled as service bound.

Since:

MHP1.0

---

getName()

```
public java.lang.String getName()
```

This method returns the name for the default language (from user properties).

If the default language does not appear in the signalling, this method will return a Name which appears in the signalling on the "best-effort basis"

Since:
MHP1.0

---

getName(String)

```
public java.lang.String getName(java.lang.String iso639Code)
```

This method returns the name of the application in the language which is specified by the parameter passed as an argument.

Throws:
LanguageNotAvailableException

Since:
MHP1.0

---

getNames()

```
public java.lang.String[][] getNames()
```

This method returns the possible names of the application, along with their ISO 639 language code. The names that are returned by this API are the names that can be found in the applicationName descriptor of the AIT. The first string in each sub array is the code.

Since:
MHP1.0

---

getPriority()

```
public int getPriority()
```

This method returns the priority field as defined in the applicationPriority field in the AIT.

Since:
MHP1.0

---

getProfiles()

```
public java.lang.String[] getProfiles()
```

This method returns an array of String, each String describing the profiles indicated in the MHP signalling as strings (ie those needed to index system properties) For implementations conforming to the 1st version of the specification, the translation will be as follows : '1' in the signalling will be translated into 'mhp.profile.enhanced_broadcast' '2' in the signalling will be translated into 'mhp.profile.interactive_broadcast'

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.

Since:
MHP1.0

getServiceLocator()

```
public org.davic.net.Locator getServiceLocator()
```

This method returns the locator of the Service describing the application.

When the transport protocol attached to the application does not indicate a 'remote connection', then the returned Locator is the Locator of the current service. Otherwise it is the service that is described in the selector field of the transport protocol descriptor.

Since:
MHP1.0

getType()

```
public int getType()
```

This method returns the type of the application (as registered by DVB).

Since:
MHP1.0

getVersions(String)

```
public int[] getVersions(java.lang.String profile)
```

This method returns an array of integers, containing the major, minor and micro values (in that order) required for the specified profile.

Throws:
IllegalProfileParameterException

Since:
MHP1.0

isStartable()

```
public boolean isStartable()
```

This method determines whether the application is startable or not. An Application is not startable when the signalling indicates that it is transmitted on a remote connection or if the caller does not have the Permissions to start it or when at the moment of the call of this method, the implementation has detected that this application is not signalled anymore.

The value returned by isStartable does not depend on whether the application is actually running or not.

Since:
MHP1.0

org.dvb.application
AppIcon

Syntax
```
public class AppIcon

java.lang.Object
  |
  +--org.dvb.application.AppIcon
```

Description
The `AppIcon` encapsulates the information concerning the icon attached to the application

Its string form is the Hex representation of the 42 bit number.

Constructors

AppIcon()

```
public AppIcon()
```

Methods

---

getIconFlags()

```
public java.util.BitSet getIconFlags()
```

extract the icon_flag fied of the application_icon_descriptor

Since:
MHP1.0

---

getLocator()

```
public org.davic.net.Locator getLocator()
```

This method returns the location of icon of the application. The Locator is constructed as follows :

when the application_type is 0x0001, the icon_locator_byte fields (as defined in the signalling) are appended to the base_directory of the application

when the application_type is 0x000, the icon_locator_byte fields (as defined in the signalling) are appended to the physical root of the application.

Since:
MHP1.0

org.dvb.application
AppID

Syntax

```
public class AppID

java.lang.Object
  |
  +--org.dvb.application.AppID
```

Description

The `AppID` is a representation of the unique identifier for applications.

Its string form is the Hex representation of the 42 bit number.

Constructors

AppID(int, int)

```
public AppID(int OID, int AID)
```

Crete a new ID based on the given Integers.

Parameters:

`OID` - the globally unique organization number.

`AID` - the unique count within the organization.

Since:

MHP1.0

Methods

---

getAID()

```
public int getAID()
```

extract the Integer value of the application count in the application name.

Since:

MHP1.0

---

getOID()

```
public int getOID()
```

extract the Integer value of the organization in the application name.

Since:

MHP1.0

toString()

```
public java.lang.String toString()
```

convert to a string containing the Hex representation of the 42 bit number.

Overrides:
java.lang.Object.toString() in class java.lang.Object

Since:
MHP1.0

org.dvb.application
AppProxy

Syntax
```
public interface AppProxy
```

All Known Subinterfaces:
DVBHTMLProxy, DVBJProxy

Description
An `AppProxy` Object is a proxy to an application. A call to the start, stop or pause will cause the resident Application Manager to respectively start, stop or pause the application bound to this AppProxy object. Each of these three method calls can throw a Security Exception if the calling application is not entitled to do so.

Each of these method call is asynchronous and will result in exactly one AppStateChangedEvent 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. See the definition of this class for more information.

Fields

DESTROYED
```
public static final int DESTROYED
```

NOT_LOADED
```
public static final int NOT_LOADED
```

PAUSED
```
public static final int PAUSED
```

STARTED
```
public static final int STARTED
```

Methods

addAppStateChangeEventListener(AppStateChangeEventListener)
```
public void addAppStateChangeEventListener(AppStateChangeEventListener 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. This method will throw a security exception if the application does not have the authority to pause the application. A call to this method will have no effect on the lifecycle of the application is the application was already in the Active state. It will have no effect if the target application is in the Paused, Destroyed or is intializing.

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

---

removeAppStateChangeEventListener(AppStateChangeEventListener)

```
public void removeAppStateChangeEventListener(AppStateChangeEventListener 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. A call to this method will have no effect on the lifecycle of the application is the application was not in the Paused state.

This method is asynchronous and its completion will be notified by an AppStateChangedEvent. In case of failure, the hasFailed method of the `AppStateChangedEvent` will return true.

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. 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, the the application will be started. In the other cases, this method call will have no effect on the lifecycle of the application.

This method is asynchronous and its completion will be notified by an AppStateChangedEvent. In case of failure, the hasFailed method of the `AppStateChangedEvent` will return true.

Throws:

`SecurityException` - if the application is not entitled to start this application.

Since:

MHP1.0

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. This method will throw a security exception if the application does not have the authority to stop the application. A call to this method will have no effect on the lifecycle of the application is the application was already in the stopped state. This method call will stop the application if it was in any other state before the call.

This method is asynchronous and its completion will be notified by an AppStateChangedEvent. In case of failure, the hasFailed method of the `AppStateChangedEvent` will return true.

Parameters:

`forced` - if true then do not ask the application but forcibly terminate it, if false give the application an oportunity to refuse.

Throws:

`SecurityException`

Since:

MHP1.0

org.dvb.application
AppsControlPermission

Syntax
```
public final class AppsControlPermission extends java.security.Permission

java.lang.Object
  |
  +--java.security.Permission
        |
        +--org.dvb.application.AppsControlPermission
```

All Implemented Interfaces:
java.security.Guard, java.io.Serializable

Description
This class represents a Permission to control the lifecyle 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 section 12.11.

AppsControlPermission(String, String)

```
public AppsControlPermission(java.lang.String s1, java.lang.String s2)
```

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 section 12.11. the parameters passed to this method have to be null.

Methods

---

checkGuard(Object)

```
public void checkGuard(java.lang.Object object)
```

method inherited from the parent class

Overrides:
java.security.Permission.checkGuard(java.lang.Object) in class java.security.Permission

Throws:
SecurityException

equals(Object)

```
public boolean equals(java.lang.Object obj)
```

Checks two AppsControlPermission objects for equality. Checks that obj is an AppsControlPermission, and has the same appid and actions as this object.

Overrides:

java.security.Permission.equals(java.lang.Object) in class java.security.Permission

getActions()

```
public java.lang.String getActions()
```

returns the list of actions that had been passed to the constructor. Inherited from the parent class. It shall return null.

Overrides:

java.security.Permission.getActions() in class java.security.Permission

hashCode()

```
public int hashCode()
```

Returns the hash code value for this object.

Overrides:

java.security.Permission.hashCode() in class java.security.Permission

implies(Permission)

```
public boolean implies(java.security.Permission permission)
```

Checks if this AppsControlPermission object "implies" the specified permission. More specifically, this method returns true if:

p is an instanceof AppsControlPermission.

Overrides:

java.security.Permission.implies(java.security.Permission) in class java.security.Permission

org.dvb.application
AppsDatabase

Syntax

```
public abstract 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 signalling. Note that the applications signalled by the external authorization descriptor 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);
       }
    }
}
```

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 available, or if the is application is signalled in the external_authorization_descriptor and is running.

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.

Parameters:

`key` - an application ID. `null` if the key is not an application ID, or not mapped to any application currently available.

Returns:

the value to which the key is mapped in this dictionary;

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 MHP 1.0, only `CurrentServiceFilter` filters may return a non empty Enumeration. If the filter object is not an instance of a subclass of `CurrentServiceFilter` then, the method shall return an empty Enumeration. When an application signalled in the external authorization descriptor is running, no AppAttribute object shall be returned for this application. 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 MHP 1.0, only `CurrentServiceFilter` filters may return a non empty Enumeration. If the filter object is not an instance of a subclass of `CurrentServiceFilter` then, the method shall return an empty Enumeration. When an application signalled in the external authorization descriptor is running, no AppAttribute object shall be returned for this application. This method will return an empty Enumeration if there are no attributes.

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. May throw a security exception if the calling application does not have the right to control applications.

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.

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;

Since:
MHP1.0

---

getAppsDatabase()

```
public static AppsDatabase getAppsDatabase()
```

Returns the singleton system-wide AppsDatabase object.

Returns:
the singleton AppsDatabase object.

Since:
MHP1.0

---

removeListener(AppsDatabaseEventListener)

```
public void removeListener(AppsDatabaseEventListener listener)
```

remove a listener on the database.

Parameters:
`listener` - the listener to be removed.

Since:
MHP1.0

---

size()

```
public int size()
```

Returns the number of applications currently available.

Returns:
the number of applications currently available.

Since:
MHP1.0

org.dvb.application
AppsDatabaseEvent

Syntax
```
public class AppsDatabaseEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.application.AppsDatabaseEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
The `AppsDatabaseEvent` class indicates either the an entry in the application database has changed, or so many changes have occurred. that the database should be considered totally new This event shall always be sent after switching to a new service. It is platform dependant if and when a new database event is thrown while tuned to the same service.

Since:
MHP1.0

Fields

APP_ADDED

```
public static final int APP_ADDED
```

The addition event id.

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.

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 an the 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 Registry object.

Since:

MHP1.0

Methods

getAppID()

```
public AppID getAppID()
```

gets the application ID object for the entry in the database that changed.

When the event type is NEW_DATABASE, AppID will be null.

Returns:

application ID representing the application

Since:

MHP1.0

getEventId()

```
public int getEventId()
```

gets the type of the event.

Returns:

an integer that matches one of the static fields describing events.

Since:

MHP1.0

org.dvb.application
AppsDatabaseEventListener

Syntax
```
public interface AppsDatabaseEventListener extends java.util.EventListener
```

All Superinterfaces:
java.util.EventListener

Description
The `AppsDatabaseListener` class allows an application to monitor the application database so that it can keep an up to date interface without polling the state. The application shall recieve 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.

Since:
MHP1.0

---

entryChanged(AppsDatabaseEvent)

```
public void entryChanged(AppsDatabaseEvent evt)
```

The AppsDataBase has had an application entry changed.

Since:
MHP1.0

---

entryRemoved(AppsDatabaseEvent)

```
public void entryRemoved(AppsDatabaseEvent evt)
```

The AppsDataBase has had an application entry removed.

Since:
MHP1.0

---

newDatabase(AppsDatabaseEvent)

```
public void newDatabase(AppsDatabaseEvent evt)
```

The AppsDataBase has radically changed.

Since:

MHP1.0

org.dvb.application
AppsDatabaseFilter

Syntax
```
public abstract class AppsDatabaseFilter

java.lang.Object
  |
  +--org.dvb.application.AppsDatabaseFilter
```

Direct Known Subclasses:
CurrentServiceFilter

Description
Abstract class for the filters. Instances of concrete classes that extend `AppsDatabaseFilter` are passed to the AppsDatabase.getAppAttributes and AppsDatabase.getAppIDs methods to allow an applications to set a filter on the list of applications (respectively AppAttributes and AppIDs) that it wants to retrieve from the AppDatabase.

For MHP 1.0, only one subclass is defined: CurrentServiceFilter

Since:
MHP 1.0

Constructors

AppsDatabaseFilter()

```
public AppsDatabaseFilter()
```

Methods

---

accept(AppID)

```
public boolean accept(AppID appid)
```

Test if a specified appid should be included in the Enumeration.

Returns:
true if the application with identifier appid should be listed.

Since:
MHP1.0

org.dvb.application
AppStateChangeEvent

Syntax
```
public class AppStateChangeEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.application.AppStateChangeEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
The `AppStateChangeEvent` class indicates a state transition of the application. If the state transition was requested by an application through this API, the method `hasFailed` indicates whether the state change failed or not. 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 be in if the state transition has succeeded.

Since:
MHP1.0

Constructors

AppStateChangeEvent(AppID, int, int, Object)

```
public AppStateChangeEvent(AppID appid, int fromstate, int tostate,
            java.lang.Object source)
```

Methods

---

getAppID()

```
public AppID getAppID()
```

The application the listener was tracking has made a state transition from `fromState` to `toState`.

Returns:
a registry entry representing the tracked application

Since:
MHP1.0

---

getFromState()

```
public int getFromState()
```

The application the listener is tracking was in `fromState`, where the value of fromState is one of the state values defined in the AppProxy interface or in the interfaces inheriting from it.

Returns:
the old state

Since:
MHP1.0

---

getToState()

```
public int getToState()
```

The application the listener is tracking is now in `toState`, where the value of toState is one of the state values defined in the AppProxy interface or in the interfaces inheriting from it.

Returns:
the new state

Since:
MHP1.0

---

hasFailed()

```
public boolean hasFailed()
```

This method determines whether an attempt to change the state of an application has failed.

Since:
MHP1.0

org.dvb.application
AppStateChangeEventListener

Syntax
```
public interface AppStateChangeEventListener extends java.util.EventListener
```

All Superinterfaces:
java.util.EventListener

Description
The `AppStateChangeEventListener` class allows a launcher application to keep track of applications it launches.

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 givent the state event.

Since:
MHP1.0

org.dvb.application
CurrentServiceFilter

Syntax
```
public class CurrentServiceFilter extends AppsDatabaseFilter

java.lang.Object
  |
  +--AppsDatabaseFilter
        |
        +--org.dvb.application.CurrentServiceFilter
```

Description
Instances of `CurrentServiceFilter` are used to set a filter on the list of applications that are retrieved from the AppsDatabase (See methods getAppsAttributes and getAppsIDs)

For MHP 1.0, only the CurrentServiceFilter class is defined A CurrentserviceFilter is used to indicate that only broadcast applications that are signalled in one of the AITs of the current service shall be returned by the getAppsAttributes and getAppIDs methods of AppsDatabase. Subclasses of CurrentServiceFilter can override the accept method so as to implement their own filter criteria on the AppID's values.

Since:
MHP 1.0

Constructors

CurrentServiceFilter()

```
public CurrentServiceFilter()
```

public Constructor of the CurrentServiceFilter

org.dvb.application
DVBHTMLProxy

Syntax
```
public interface DVBHTMLProxy extends AppProxy
```

All Superinterfaces:
AppProxy

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.

Since:
MHP1.0

---

startTrigger(Date)

```
public void startTrigger(java.util.Date starttime)
```

sends the application a start trigger at the indicated time. 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.

Since:
MHP1.0

---

trigger(Date, Object)

```
public void trigger(java.util.Date time, java.lang.Object triggerPayload)
```

sends the application a trigger at the indicated time. If the time has already passed the application manager should send the trigger immediatelyt. Dates pre-epoch shall always cause the application manager to send a 'now' trigger. The payload is specified as object, but this will be refined once DVB-HTML Triggers are properly defined.

Since:
MHP1.0

org.dvb.application
DVBJProxy

Syntax
```
public interface DVBJProxy extends AppProxy
```

All Superinterfaces:
AppProxy

Fields

INITED
```
public static final int INITED
```

LOADED
```
public static final int LOADED
```

Methods

init()
```
public void init()
```

Requests the application manager calls the Xlet init method on the application

This method is asynchronous and its completion will be notified by an AppStateChangedEvent. In case of failure, the hasFailed method of the `AppStateChangedEvent` will return true. A call to this method will have no effect on the lifecycle of the application is the application was already in the paused 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

load()
```
public void load()
```

Loads the classes of the application on a best effort basis

This method is asynchronous and its completion will be notified by an AppStateChangedEvent. In case of failure, the hasFailed method of the `AppStateChangedEvent` will return true. A call to this method will have no effect on the lifecycle of the application is the application was already in the

Paused state. In all cases, an AppStateChangeEvent will be sent, whether the call was successful or not.

Throws:

`SecurityException` - if the application is not entitled to load this application. being able to load an application requires to be entitled to start it.

Since:

MHP1.0

org.dvb.application
IllegalProfileParameterException

Syntax
```
public class IllegalProfileParameterException extends java.lang.Exception

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.dvb.application.IllegalProfileParameterException
```

All Implemented Interfaces:
java.io.Serializable

Description
The `IllegalProfileParameter` exception is thrown if the application attempts to ask for a version number for a profile not specified for the application.

Since:
MHP1.0

Constructors

IllegalProfileParameterException()

```
public IllegalProfileParameterException()
```

org.dvb.application
LanguageNotAvailableException

Syntax
```
public class LanguageNotAvailableException extends java.lang.Exception

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.dvb.application.LanguageNotAvailableException
```

All Implemented Interfaces:
java.io.Serializable

Description
The `LanguageNotAvailableException` exception is thrown if the application asks for the name of an application in a language not signalled in the AIT.

Since:
MHP1.0

Constructors

LanguageNotAvailableException()

```
public LanguageNotAvailableException()
```

org.dvb.application
ProxyInUseException

Syntax
```
public class ProxyInUseException extends java.lang.Exception

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.dvb.application.ProxyInUseException
```

All Implemented Interfaces:
java.io.Serializable

Description
The `ProxyInUseException` exception is thrown if the application attempts to alter the registration for an application but another application is currently holding a proxy to the same application, or the application is running

Since:
MHP1.0

Constructors

ProxyInUseException()

```
public ProxyInUseException()
```

Annex T (normative): Permissions

Package
org.dvb.net.ca

Class Summary	
Classes	
CAPermission	This class is for CA permissions.

org.dvb.net.ca
CAPermission

Syntax

```
public class CAPermission extends java.security.BasicPermission

java.lang.Object
  |
  +--java.security.Permission
        |
        +--java.security.BasicPermission
              |
              +--org.dvb.net.ca.CAPermission
```

All Implemented Interfaces:

java.security.Guard, java.io.Serializable

Description

This class is for CA permissions. A CAPermission contains a name, but no actions list.

A CAPermission contains a range of CA system ids and a specific permission for that range of CA system ids. Instead of a range of CA system ids, the CAPermission can also refer to a single CA system id.

The name has the following syntax:

CASystemIdRange ":" Permission

where CASystemIdRange = CASystemId ["-" CASystemId] | "*"

and Permission = "MMI" | "buy" | "entitlementInfo" | "messagePassing" | "*"

Examples:

- 0x1200-0x120A:buy (The permission to buy entitlement for all the CA systems with ids between 0x1200 and 0x120A inclusive.)
- 1201:entitlementInfo (The permission to get entitlement information for the CA system with id 1201)
- 0x120d:* (This wildcard expresses all the permissions for the CA system with id 0x120d).

Note: The CASystemId is expressed as a hexadecimal value.

The permission "MMI" corresponds with the SecurityException on CAModuleManager.addMMIListener(). The permission "buy" corresponds with the SecurityException on CAModule.buyEntitlement(). The permission "entitlementInfo" corresponds with the SecurityException on CAModule.queryEntitlement() and CAModule.listEntitlements(). The permission "messagePassing" corresponds with CAModule.openMessageSession(MessageListener)

Constructors

CAPermission(String)

```
public CAPermission(java.lang.String name)
```

Creates a new CAPermission with the specified name. The name is the symbolic name of the CAPermission.

Parameters:

`name` - the name of the CAPermission

CAPermission(String, String)

```
public CAPermission(java.lang.String name, java.lang.String actions)
```

Creates a new CAPermission object with the specified name. The name is the symbolic name of the CAPermission, and the actions String is unused and should be null. This constructor exists for use by the Policy object to instantiate new Permission objects.

Parameters:

`name` - the name of the CAPermission

`actions` - should be null.

Package
org.dvb.net.tuning

Class Summary	
Classes	
TunerPermission	This class is for tuner permissions.

org.dvb.net.tuning
TunerPermission

Syntax
```
public class TunerPermission extends java.security.BasicPermission

java.lang.Object
  |
  +--java.security.Permission
        |
        +--java.security.BasicPermission
              |
              +--org.dvb.net.tuning.TunerPermission
```

All Implemented Interfaces:
java.security.Guard, java.io.Serializable

Description
This class is for tuner permissions. A TunerPermission contains no name and no actions list. If an application has the tuner permission, then it is allowed to tune using the Tuning API.

Constructors

TunerPermission(String)

```
public TunerPermission(java.lang.String name)
```

Creates a new TunerPermission. The name parameter is not used.

Parameters:

`name` - the name of the TunerPermission. Not used, shall be "".

TunerPermission(String, String)

```
public TunerPermission(java.lang.String name, java.lang.String actions)
```

Creates a new TunerPermission. The name and actions parameters are not used. This constructor exists for use by the Policy object to instantiate new Permission objects.

Parameters:

`name` - the name of the CAPermission. Not used, shall be "".

`actions` - should be null.

Annex U (normative): Extended graphics APIs

Package org.dvb.ui

Class Summary

Interfaces

TestOpacity	Interface implemented by Components or Containers in order to allow the platform to query whether their paint method is fully opaque.
TextOverflowListener	The TextOverflowListener is an interface that an application may implement and register in the DVBTextLayoutManager.

Classes

DVBAlphaComposite	This DVBAlphaComposite class implements the basic alpha compositing rules for combining source and destination pixels to achieve blending and transparency effects with graphics, images and video.
DVBBufferedImage	The DVBBufferedImage subclass describes an java.awt.Image with an accessible buffer of image data.
DVBColor	A Color class which adds the notion of alpha.
DVBGraphics	The DVBGraphics class is a adapter class to support alpha compositing in an MHP device.
DVBTextLayoutManager	The DVBTextLayoutManager provides a text rendering layout mechanism for the org.havi.ui.HStaticText org.havi.ui.HText and org.havi.ui.HTextButton classes.

Exceptions

UnsupportedDrawingOperationException	The UnsupportedDrawingOperationException class represents an exception that is thrown if an drawing operation is not supported on this platform.

org.dvb.ui
DVBAlphaComposite

Syntax

```
public final class DVBAlphaComposite

java.lang.Object
  |
  +--org.dvb.ui.DVBAlphaComposite
```

Description

This `DVBAlphaComposite` 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:

- Cs = one of the color components of the source pixel.
- Cd = one of the color components of the destination pixel.
- As = alpha component of the source pixel.
- Ad = alpha component of the destination pixel.
- Fs = fraction of the source pixel that contributes to the output.
- Fd = 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:

```
Cd = Cs*Fs + Cd*Fd
Ad = As*Fs + Ad*Fd
```

where Fs and Fd are specified by each rule. The above equations assume that both source and destination pixels have the color components premultiplied by the alpha component. Similarly, the equations expressed in the definitions of compositing rules below assume premultiplied alpha.

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.

Since:

MHP 1.0

Fields

Clear

```
public static final DVBAlphaComposite Clear
```

`DVBAlphaComposite` object that implements the opaque CLEAR rule with an alpha of 1.0f.

Since:

MHP 1.0

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.

Fs = 0 and Fd = 0, thus:

```
Cd = 0
Ad = 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

Since:
MHP 1.0

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.

Fs = 0 and Fd = As, thus:

```
Cd = Cd*As
Ad = Ad*As
```

Note that this operation is faster than e.g. SRC_OVER but slower as SRC

Since:
MHP 1.0

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.

Fs = 0 and Fd = (1-As), thus:

```
Cd = Cd*(1-As)
Ad = Ad*(1-As)
```

Note that this operation is faster than e.g. SRC_OVER but slower as SRC

Since:
MHP 1.0

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.

Fs = (1-Ad) and Fd = 1, thus:

```
Cd = Cs*(1-Ad) + Cd
Ad = As*(1-Ad) + Ad
```

Note that this can be a very slow drawing operation

Since:
MHP 1.0

DstIn

```
public static final DVBAlphaComposite DstIn
```

DVBAlphaComposite object that implements the opaque DST_IN rule with an alpha of 1.0f.

Since:
MHP 1.0

See Also:
DST_IN

DstOut

```
public static final DVBAlphaComposite DstOut
```

DVBAlphaComposite object that implements the opaque DST_OUT rule with an alpha of 1.0f.

Since:
MHP 1.0

See Also:
DST_OUT

DstOver

```
public static final DVBAlphaComposite DstOver
```

DVBAlphaComposite object that implements the opaque DST_OVER rule with an alpha of 1.0f.

Since:
MHP 1.0

See Also:
DST_OVER

Src

```
public static final DVBAlphaComposite Src
```

DVBAlphaComposite object that implements the opaque SRC rule with an alpha of 1.0f.

Since:
MHP 1.0

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:

```
Cd = Cs
Ad = As
```

Note that this is a fast drawing routine

Since:
MHP 1.0

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:

```
Cd = Cs*Ad
Ad = As*Ad
```

Note that this operation is faster than e.g. SRC_OVER but slower as SRC

Since:
MHP 1.0

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.

Fs = (1-Ad) and Fd = 0, thus:

```
Cd = Cs*(1-Ad)
Ad = As*(1-Ad)
```

Note that this operation is faster than e.g. SRC_OVER but slower as SRC

Since:
MHP 1.0

SRC_OVER

```
public static final int SRC_OVER
```

Porter-Duff Source Over Destination rule. The source is composited over the destination.

Fs = 1 and Fd = (1-As), thus:

```
Cd = Cs + Cd*(1-As)
Ad = As + Ad*(1-As)
```

Note that this can be a very slow drawing operation

Since:
MHP 1.0

SrcIn

```
public static final DVBAlphaComposite SrcIn
```

`DVBAlphaComposite` object that implements the opaque SRC_IN rule with an alpha of 1.0f.

Since:
MHP 1.0

See Also:
SRC_IN

SrcOut

```
public static final DVBAlphaComposite SrcOut
```

`DVBAlphaComposite` object that implements the opaque SRC_OUT rule with an alpha of 1.0f.

Since:

MHP 1.0

See Also:
SRC_OUT

SrcOver

```
public static final DVBAlphaComposite SrcOver
```

`DVBAlphaComposite` object that implements the opaque SRC_OVER rule with an alpha of 1.0f.

Since:
MHP 1.0

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 `DVBAlphaComposite` 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` equals this `DVBAlphaComposite`; `false` otherwise.

Since:
MHP 1.0

---

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`.

Since:
MHP 1.0

---

getInstance(int)

```
public static DVBAlphaComposite getInstance(int rule)
```

Creates an `DVBAlphaComposite` object with the specified rule.

Parameters:
`rule` - the compositing rule

Since:
MHP 1.0

---

getInstance(int, float)

```
public static DVBAlphaComposite getInstance(int rule, float alpha)
```

Creates an `DVBAlphaComposite` object with the specified rule and the constant alpha to multiply with the alpha of the source. The source is multiplied with the specified alpha before being composited with the destination.

Parameters:

`rule` - the compositing rule

`alpha` - the constant alpha to be multiplied with the alpha of the source. `alpha` must be a floating point number in the inclusive range [0.0, 1.0].

Since:

MHP 1.0

---

getRule()

```
public int getRule()
```

Returns the compositing rule of this `DVBAlphaComposite`.

Returns:

the compositing rule of this `DVBAlphaComposite`.

Since:

MHP 1.0

org.dvb.ui
DVBBufferedImage

Syntax

```
public class DVBBufferedImage extends java.awt.Image implements java.lang.Cloneable

java.lang.Object
  |
  +--java.awt.Image
        |
        +--org.dvb.ui.DVBBufferedImage
```

All Implemented Interfaces:

java.lang.Cloneable

Description

The `DVBBufferedImage` subclass describes an `java.awt.Image` with an accessible buffer of image data. The DVBBufferedImage is an adapter class for java.awt.image.BufferedImage. It supports two different platform dependent sample models TYPE_BASE and TYPE_ADVANCED. Buffered images with the TYPE_BASE have the same sample model as the on screen graphics buffer, thus TYPE_BASE could be CLUT based. TYPE_ADVANCED has a direct color model but it is not specified how many bits are used to store the different color components. By default, a new DVBBufferedImage is transparent. All alpha values are set to 0;

Since:

MHP 1.0

Fields

TYPE_ADVANCED

```
public static final int TYPE_ADVANCED
```

Represents an image stored in a best possible SampleModel (platform dependent) The image has a DirectColorModel with alpha. The color data in this image is considered not to be premultiplied with alpha. The data returned by getRGB() will be in the TYPE_INT_ARGB color model that is alpha component in bits 24-31, the red component in bits 16-23, the green component in bits 8-15, and the blue component in bits 0-7. The data for setRGB() shall be in the TYPE_INT_ARGB color model as well.

Since:

MHP 1.0

TYPE_BASE

```
public static final int TYPE_BASE
```

Represents an image stored in a platform dependent Sample Model. This color model is not visible to applications. The data returned by getRGB() will be in the TYPE_INT_ARGB color model that is alpha component in bits 24-31, the red component in bits 16-23, the green component in bits 8-15, and the blue component in bits 0-7. The data for setRGB() shall be in the TYPE_INT_ARGB color model as well.

Since:
MHP 1.0

Constructors

DVBBufferedImage(int, int)

```
public DVBBufferedImage(int width, int height)
```

Constructs a DVBBufferedImage with the specified width and height. The Sample Model used the image is the native Sample Model (TYPE_BASE) of the implementation. Note that a request can lead to an java.lang.OutOfMemoryError. Applications should be aware of this.

Parameters:
`width` - - width of the created image

Since:
MHP 1.0

DVBBufferedImage(int, int, int)

```
public DVBBufferedImage(int width, int height, int typ)
```

Constructs a new DVBBufferedImage with the specified width and height in the Sample Model specified by typ. Note that a request can lead to an java.lang.OutOfMemoryError. Applications should be aware of this.

Parameters:
`width` - - the width of the DVBBufferedImage

`height` - - the height of the DVBBufferedImage

`type` - - the ColorSpace of the DVBBufferedImage

Since:
MHP 1.0

Methods

createGraphics()

```
public DVBGraphics createGraphics()
```

Creates a `DVBGraphics`, which can be used to draw into this `DVBBufferedImage`.

Returns:
a `DVBGraphics`, used for drawing into this image.

Since:
MHP 1.0

flush()

```
public void flush()
```

Flushes all resources being used to cache optimization information. The underlying pixel data is unaffected.

Overrides:
java.awt.Image.flush() in class java.awt.Image

---

getGraphics()

```
public java.awt.Graphics getGraphics()
```

This method returns a `java.awt.Graphics`, it is here for backwards compatibility. createGraphics() is more convenient, since it is declared to return a `DVBGraphics`.

Overrides:
java.awt.Image.getGraphics() in class java.awt.Image

Returns:
a `Graphics`, which can be used to draw into this image.

---

getHeight()

```
public int getHeight()
```

Returns the height of the `DVBBufferedImage`.

Returns:
the height of this `DVBBufferdImage`.

Since:
MHP 1.0

---

getHeight(ImageObserver)

```
public int getHeight(java.awt.image.ImageObserver observer)
```

Returns the actual height of the image. If the height is not known yet then the `ImageObserver` is notified later and `-1` is returned.

Overrides:
java.awt.Image.getHeight(java.awt.image.ImageObserver) in class java.awt.Image

Parameters:
`observer` - the `ImageObserver` that receives information about the image

Returns:
the height of the image or `-1` if the height is not yet known.

See Also:
`java.awt.Image.getWidth(ImageObserver)`, `java.awt.image.ImageObserver`

---

getImage()

```
public java.awt.Image getImage()
```

In implementations using the JDK1.2 API this returns an BufferedImage cast to a java.awt.Image, in other implementations it returns this DVBBufferedImage as an image.

Since:
MHP 1.0

getProperty(String, ImageObserver)

```
public java.lang.Object getProperty(java.lang.String name,
            java.awt.image.ImageObserver observer)
```

Returns a property of the image by name. Individual property names are defined by the various image formats. If a property is not defined for a particular image, this method returns the `UndefinedProperty` field. If the properties for this image are not yet known, then this method returns `null` and the `ImageObserver` object is notified later. The property name "comment" should be used to store an optional comment that can be presented to the user as a description of the image, its source, or its author.

Overrides:

java.awt.Image.getProperty(java.lang.String, java.awt.image.ImageObserver) in class java.awt.Image

Parameters:

`name` - the property name

`observer` - the `ImageObserver` that receives notification regarding image information

Returns:

an `java.lang.Object` that is the property referred to by the specified `name` or `null` if the properties of this image are not yet known.

See Also:

`java.awt.image.ImageObserver`, `java.awt.Image.UndefinedProperty`

getRGB(int, int)

```
public int getRGB(int x, int y)
```

Returns an integer pixel in the default RGB color model 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 whan a lower precission is used in this buffered image getRGB may return different values than those used in setRGB()

Parameters:

`x, y` - the coordinates of the pixel from which to get the pixel in the default RGB color model and sRGB color space

Returns:

an integer pixel in the default RGB color model and default sRGB colorspace.

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 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 image, the ARGB pixel can be accessed in this way:

```
pixel   = rgbArray[offset + (y-startY)*scansize + (x-startX)];
```

Parameters:

`startX, ` - startY the starting coordinates

`w` - width of region

`h` - height of 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 RGB pixels.

Throws:
`<code>IllegalArgumentException</code>` - if an unknown datatype is specified

Since:
MHP 1.0

---

getSource()

```
public java.awt.image.ImageProducer getSource()
```

Returns the object that produces the pixels for the image.

Overrides:
java.awt.Image.getSource() in class java.awt.Image

Returns:
the `java.awt.image.ImageProducer` that is used to produce the pixels for this image.

See Also:
`java.awt.image.ImageProducer`

---

getSubimage(int, int, int, int)

```
public DVBBufferedImage getSubimage(int x, int y, int w, int h)
```

Returns a subimage defined by a specified rectangular region. The returned `DVBBufferedImage` shares the same data array as the original image.

Parameters:
`x, y` - the coordinates 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 `DVBBufferdImage` that is the subimage of this `DVBBufferdImage`.

Throws:
`<code>RasterFormatException</code>` - if the specified area is not contained within this `DVBBufferdImage`.

Since:
MHP 1.0

---

getWidth()

```
public int getWidth()
```

Returns the width of the `DVBBufferedImage`.

Returns:
the width of this `DVBBufferdImage`.

Since:
MHP 1.0

---

getWidth(ImageObserver)

```
public int getWidth(java.awt.image.ImageObserver observer)
```

Returns the actual width of the image. If the width is not known yet then the `java.awt.image.ImageObserver` is notified later and `-1` is returned.

Overrides:
java.awt.Image.getWidth(java.awt.image.ImageObserver) in class java.awt.Image

Parameters:
`observer` - the `ImageObserver` that receives information about the image

Returns:
the width of the image or `-1` if the width is not yet known.

See Also:
`java.awt.Image.getHeight(ImageObserver)`, `java.awt.image.ImageObserver`

---

setRGB(int, int, int)

```
public synchronized void setRGB(int x, int y, int rgb)
```

Sets a pixel in this `DVBBufferedImage` to the specified RGB value. The pixel is assumed to be in the default RGB color model, TYPE_INT_ARGB, and default sRGB color space.

Parameters:
`x, y` - the coordinates of the pixel to set

`rgb` - the RGB 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 portion of the image data. There are only 8-bits of precision for each color component in the returned data when using this method. With a specified coordinate (x, y) in the this image, the ARGB pixel can be accessed in this way:

```
pixel   = rgbArray[offset + (y-startY)*scansize + (x-startX)];
```

WARNING: No dithering takes place.

Parameters:
`startX, startY` - the starting coordinates

`w` - width of the region

`h` - height of the region

`rgbArray` - the rgb 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 `DVBBufferdImage`.

org.dvb.ui
DVBColor

Syntax

```
public class DVBColor extends org.davic.awt.Color

java.lang.Object
  |
  +--java.awt.Color
        |
        +--org.davic.awt.Color
              |
              +--org.dvb.ui.DVBColor
```

All Implemented Interfaces:

java.io.Serializable

Description

A Color class which adds the notion of alpha. It is compatible with the JDK1.2 java.awt.Color class DVB-Color extends org.davic.awt.Color which extends java.awt.Color. In implementations using the JDK1.1 this class adds support for alpha In implementations using the JDK1.2 the additional methods would just call super. Because DVBColor extends Color the signatures in the existing classes do not change. Classes like Component should work with DVBColor internaly.

Since:

MHP 1.0

Constructors

DVBColor(Color)

```
public DVBColor(java.awt.Color c)
```

Constructs a new DVBColor using the specified java.awt.Color. If c is a JDK1.1 color alpha will be set to 1.0 (opaque), if c is a JDK1.2 color the alpha of c will be used.

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 componentg - the green componentb - the blue componenta - 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 componentshasalpha - 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 componentg - the green componentb - the blue componenta - 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:

java.awt.Color.brighter() in class java.awt.Color

Returns:

a new DVBColor object cast to a java.awt.Color Object, Applications can recast it to a org.dvb.ui.DVBColor Object a brighter version of this color.

Since:

JDK1.0

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:
java.awt.Color.darker() in class java.awt.Color

Returns:
a new DVBColor object cast to a java.awt.Color Object, Applications can recast it to a org.dvb.ui.DVBColor Object a darker version of this color.

Since:
JDK1.0

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:
java.awt.Color.equals(java.lang.Object) in class java.awt.Color

Parameters:
`obj` - - the object to compare with.

Returns:
true if the objects are the same; false otherwise.

Since:
JDK1.0

---

getAlpha()

```
public int getAlpha()
```

Returns the alpha component. In the range 0-255.

Overrides:
org.davic.awt.Color.getAlpha() in class org.davic.awt.Color

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:
java.awt.Color.getRGB() in class java.awt.Color

Since:
JDK1.0

See Also:
`java.awt.Color.getRed()`, `java.awt.Color.getGreen()`, `java.awt.Color.getBlue()`, getAlpha()

toString()

```
public java.lang.String toString()
```

Creates a string that represents this color and indicates the values of its ARGB components.

Overrides:
java.awt.Color.toString() in class java.awt.Color

Returns:
a representation of this color as a String object.

Since:
JDK1.0

org.dvb.ui
DVBGraphics

Syntax

```
public abstract class DVBGraphics extends java.awt.Graphics

java.lang.Object
  |
  +--java.awt.Graphics
        |
        +--org.dvb.ui.DVBGraphics
```

Description

The `DVBGraphics` class is a adapter class to support alpha compositing in an MHP device. Most methods directly delegate to java.awt.Graphics other methods could delegate to the appropriate methods in java.awt.Graphics2D where available or could be implemented in native code This class inherits from java.awt.Graphics in implementations using the JDK1.1. In implementations using the JDK1.2 DVBGraphics inherits from java.awt.Graphics2D. **In MHP devices all Graphics Objects are DVBGraphics objects.** Thus one can get a DVBGraphics by casting a given Graphics object. The normal compositing rule used is **DVBAlphaComposite.SCR. This is the fastes rule because there is no computation. Note this is not the default behaviour of other graphics libraries. When drawing pictures with an alpha channel the transparent part will be transparent to the video in the background. Programmers should set the rule to DVBAlphaComposite.SRC_OVER when drawing images or other shapes which shall be transparent to the graphics in the background**

Since:

MHP1.0

See Also:

`java.awt.Graphics`

Constructors

DVBGraphics()

```
protected DVBGraphics()
```

Constructs a new `DVBGraphics` object. This constructor is the default contructor 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 DVBAlphaComposite[] getAvailableCompositeRules()
```

Returns all available Porter-Duff Rules for this specific Graphics context. E.g. a devices could support the SRC_OVER rule when using a destination which does not has Alpha or were the alpha is null while this rule is not available when drawing on a graphic context were the destination has alpha. Which rules are supported for the different graphics objects is defined in the Minimum Platform Capabilities of the MHP spec.

Since:
MHP 1.0

getBestColorMatch(Color)

```
public DVBColor getBestColorMatch(java.awt.Color c)
```

Returns the best match for the specified Color as a DVBColor

Returns:
- the best 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 DVBAlphaComposite getDVBComposite()
```

Returns the current `DVBAlphaComposite` in the `DVBGraphics` context. This method could delegate to a java.awt.Graphics2D object where available

Returns:
the current `DVBGraphics` `DVBAlphaComposite`, which defines a compositing style.

Since:
MHP 1.0

See Also:
setDVBComposite(DVBAlphaComposite)

getType()

```
public int getType()
```

Returns the Sample Model (DVBBufferedImage.TYPE_BASE, DVBBufferedImage.TYPE_ADVANCED) which is used in the on/off screen buffer this graphics object draws into.

Returns:
the type of the Sample Model

Since:
MHP 1.0

See Also:
DVBBufferedImage

setColor(Color)

```
public abstract void setColor(java.awt.Color c)
```

Sets this graphics context's current color to the specified color. All subsequent graphics operations using this graphics context use this specified color. Note that color c can be a DVBColor

Overrides:
java.awt.Graphics.setColor(java.awt.Color) in class java.awt.Graphics

Parameters:
`c` - the new rendering color.

Since:
MHP 1.0

See Also:
`java.awt.Color`, DVBColor, getColor()

setDVBComposite(DVBAlphaComposite)

```
public abstract void setDVBComposite(DVBAlphaComposite comp)
```

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

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
DVBTextLayoutManager

Syntax

```
public class DVBTextLayoutManager implements HTextLayoutManager

java.lang.Object
  |
  +--org.dvb.ui.DVBTextLayoutManager
```

All Implemented Interfaces:

HTextLayoutManager

Description

The DVBTextLayoutManager provides a text rendering layout mechanism for the org.havi.ui.HStaticText org.havi.ui.HText and org.havi.ui.HTextButton classes.

The semantics of the rendering behaviour and the settings are specified in Annex C of the DVB MHP specification. The DVBTextLayoutManager renders the text according to the semantics described in Annex C.

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 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 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 vertically to the 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 right).

VERTICAL_START_ALIGN

```
public static final int VERTICAL_START_ALIGN
```

The text should be aligned vertically to the 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).

Constructors

DVBTextLayoutManager()

```
public DVBTextLayoutManager()
```

Constructs a DVBTextLayoutManager object with default parameters (HORIZONTAL_START_ALIGN, LINE_ORIENTATION_HORIZONTAL, START_CORNER_UPPER_RIGHT, 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.

---

getInsets()

```
public java.awt.Insets getInsets()
```

Returns the insets that this text layout manager uses. When rendering text, it leaves empty margins of the size defined by the insets. The actual area used for the text is the area of the component decreased by the mount of insets at each edge. The deafult insets, if not set explicitly useing setInsets, are 0 at each edge, i.e. no margins.

---

getLetterSpace()

```
public int getLetterSpace()
```

Get the letter space setting.

Returns:
letter space setting

---

getLineOrientation()

```
public int getLineOrientation()
```

Get the line orientation.

Returns:
Line orientation setting

---

getLineSpace()

```
public int getLineSpace()
```

Get the line space setting.

Returns:
line space setting

---

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)

```
public void render(java.lang.String markedUpString, java.awt.Graphics g, HVisible v)
```

The DVBTextLayoutManager uses the java.awt.Graphics object and org.havi.ui.HVisible as a basis to determine the rendering. The DVBTextLayoutManager uses the Font set in the HVisible object as the default font for the text. It also uses the Color set as foreground colour in the HVisible for the default text colour.

Specified By:
render(String, Graphics, HVisible) in interface HTextLayoutManager

Parameters:
`markedUpString` - the marked-up string to parse, and render. The format of the string and the control codes that can be used are specified in Annex C of the DVB MHP specification.

`g` - the java.awt.Graphics object whose properties are used as a basis for the rendering.

`v` - the HVisible object whose properties are used as a basis for the rendering.

See Also:
HTextLayoutManager

---

setHorizontalAlign(int)

```
public void setHorizontalAlign(int horizontalAlign)
```

Set the horizontal alignment.

Parameters:
`horizontalAlign` - Horizontal alignment setting

---

setHorizontalTabSpacing(int)

```
public void setHorizontalTabSpacing(int horizontalTabSpace)
```

Set the horizontal tabulation spacing.

Parameters:
`horizontalTabSpace` - tab spacing in points

---

setInsets(Insets)

```
public void setInsets(java.awt.Insets insets)
```

Sets the insets that should be used by this text layout manager. The text is rendered to the area defined by the area of the component decreased by the amount of insets at each edge. If this method is not called, the default insets are 0 at each edge.

Parameters:
`insets` - Insets that should be used

---

setLetterSpace(int)

```
public void setLetterSpace(int letterSpace)
```

Set the letter space setting.

Parameters:
`letterSpace` - letter space setting

---

setLineOrientation(int)

```
public void setLineOrientation(int lineOrientation)
```

Set the line orientation.

Parameters:
`lineOrientation` - Line orientation setting

---

setLineSpace(int)

```
public void setLineSpace(int lineSpace)
```

Set the line space setting.

Parameters:
`lineSpace` - line space setting

---

setStartCorner(int)

```
public void setStartCorner(int startCorner)
```

Set the starting corner.

Parameters:
`startCorner` - Starting corner setting

---

setTextWrapping(boolean)

```
public void setTextWrapping(boolean wrap)
```

Set the text wrapping setting

Parameters:
`wrap` - Text wrapping setting

---

setVerticalAlign(int)

```
public void setVerticalAlign(int verticalAlign)
```

Set the vertical alignment.

Parameters:
`verticalAlign` - Vertical alignment setting

org.dvb.ui
TestOpacity

Syntax
```
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) guarantee that all pixels are painted in an opaque Color.

By default, the return value is false. The return value should be overridden by subclasses that can guarantee full opacity. The consequences of an invalid overridden value are implementation specific.

Returns:

true if all the pixels with the java.awt.Component#getBounds method are fully opaque, otherwise false.

org.dvb.ui
TextOverflowListener

Syntax
```
public interface TextOverflowListener
```

Description
The TextOverflowListener is an interface that an application may implement and register in the DVBTextLayoutManager. This listener will be notified if the text string does not fit within the component when rendering it.

Methods

---

notifyTextOverflow(String, HVisible, boolean, boolean)

```
public void notifyTextOverflow(java.lang.String markedUpString, HVisible v,
            boolean overflowedHorizontally, boolean overflowedVertically)
```

This method is called by the DVBTextLayoutManager if the text does not fit within the component

Parameters:

`markedUpString` - the string that was rendered

`v` - the HVisible object that was being rendered

`overflowedHorizontally` - true if the text overflew the bounds of the component in the horizontal direction; otherwise false

`overflowedVertically` - true if the text overflew the bounds of the component in the vertical direction; otherwise false

org.dvb.ui

UnsupportedDrawingOperationException

Syntax

```
public class UnsupportedDrawingOperationException extends java.lang.Exception

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.dvb.ui.UnsupportedDrawingOperationException
```

All Implemented Interfaces:

java.io.Serializable

Description

The `UnsupportedDrawingOperationException` class represents an exception that is thrown if an drawing operation is not supported on this platform. E.g. DVBGraphics.setComposite could throw an Exception when setting the SRC_OVER rule on some devices while the SRC rule will always work.

Since:

MHP 1.0

Constructors

UnsupportedDrawingOperationException(String)

```
public UnsupportedDrawingOperationException(java.lang.String s)
```

Constructs an instance of `UnsupportedDrawingOperationException` with the specified detail message.

Parameters:

`s` - the detail message

Since:

MHP1.0

Annex V (normative): HAVi Level 2 User Interface

havi.ui

Sony Matsushita
Philips Thomson
Hitachi Toshiba
Sharp Grundig

Version 1.0
January 18, 2000

1 HAVi Level 2 User Interface

This chapter provides a specification of the Home Audio/Video Interoperability Architecture User-Interface, (also called the HAVi User-Interface). This HAVi User-Interface is designed as a "TV-friendly" user-interface framework and is explicitly designed to be suitable for use and implementation on a variety of consumer electronic (CE) devices. The application programming interfaces (APIs) for the HAVi User-Interface are contained in the org.havi.ui and org.havi.ui.event packages described in Appendix A: HAVi Java APIs.

1.1 HAVi User-Interface Design

The HAVi User-Interface allows applications, written in Java, to determine the user interface capabilities of its host display device, accept input from the user, draw to the screen and play audio clips. It uses the JDK 1.1 java.awt package lightweight component framework as its core and extends this with packages and classes specific to the HAVi platform.

1.1.1 Remote Control

The user input model from java.awt is extended to support an optional remote control. A large number of events are optional, allowing manufacturers to customize and add value to their products.

1.1.2 Television Specific Support

HAVi also adds classes to support graphics and video display functions that are available in typical television-based systems, including: support for non-square pixels, and graphics / video overlays.

1.1.3 Widgets

The HAVi User-Interface augments the java.awt "Lightweight User Interface Framework" with an abstract widget framework and a set of HAVi classes derived from this, which represent TV-friendly widgets that are *guaranteed* to be available to HAVi applications not wishing to implement their own custom widgets.

Alternatively, applications can derive custom widgets by subclassing the HAVi base widgets, using the abstract widget framework, or by employing Java's "Lightweight User Interface Framework".

1.2 java.awt Subset

Only a subset of the standard JDK1.1 java.awt package is required to be present on a HAVi platform. The required subset of java.awt includes all of the classes needed to support the "Lightweight User Interface Framework", which allows application developers to implement their own user-interface widgets. HAVi applications which rely on the presence of the complete JDK 1.1 java.awt package cannot be considered as interoperable with all HAVi implementations.

1.2.1 Motivation

The user interface components supplied with early versions of Java – which are still present in the

java.awt package in JDK 1.1 – are normally implemented using native methods to provide a platform specific look and feel. These components are implemented using native code. As such, they can be implemented to offer high efficiency to the microprocessor. The architecture of the API reflects this in a package called java.awt.peer. The "peer" classes in this package provide the interface between the Java and the normally native code. The disadvantages of these existing components are that they take twice as many classes to implement – a Java class, plus an associated peer class – and that they are not able to create transparent, non-rectangular shapes.

"Lightweight components", on the other hand, have no direct peer – they are required to implement their own look and feel using Java code. Each lightweight widget is created by subclassing the java.awt.Component class. Because lightweight components lack an associated peer class, they must handle their own presentation, which they do by redefining a Java method called "paint".

The application developer has two choices: use the resident widgets or extend the java.awt.Component and java.awt.Container classes to provide a fully customized look and feel. The first approach provides some flexibility while retaining a compact application size. The latter approach allows the greatest flexibility, but with extra complexity in the application.

1.2.2 Required Elements from AWT

Since most of the widget set in the java.awt package is not "TV friendly", these classes are not required to be present in systems supporting the HAVi UI framework. TV friendly equivalents of these are provided through the HAVi User-Interface framework, which can be extended to support alternative look and feel. Thus some of the standard JDK 1.1 java.awt package classes, including some widgets (as defined by those parts of the java.awt package not included in this specification) are not guaranteed to be present in all devices supporting the HAVi User-Interface framework. Applications wishing to be interoperable with all devices supporting the HAVi User-Interface should not make use of these classes. Where an application uses classes which fall outside of the scope of the HAVi specification, this behavior is not determined by this HAVi User-Interface specification, rather it shall be determined by the implementation of the underlying platform, for example, the failure mode on a JDK-based implementation, may be dissimilar from the failure mode for a Personal Java, or Embedded Java-based implementation. This specification does not prevent a manufacturer implementing a particular device using all of AWT, and any applications intended to execute solely in a particular device may of course exploit any classes or packages known to be in that device, but both the device and application should not be regarded as interoperable and should be considered to be proprietary in nature.

The specified set of classes have been chosen such that HAVi applications can implement any missing widget functionality using these classes (lightweight components).

- The main base classes, such as java.awt.Component, are required in order to build the lightweight components.
- Other classes, such as java.awt.Color and java.awt.Font, are required for all general drawing and painting.
- The layout classes, such as java.awt.FlowLayout and java.awt.BorderLayout are retained to provide flexible layout of components on various output devices.

The classes from java.awt that are listed in Table 1 are the classes that an HAVi application author can reliably interact with, and use within a HAVi compliant application. Interoperable applications must not use any references from classes in this list to classes not in this list.

Table 1. java.awt Classes Available to Interoperable HAVi Applications

java.awt	java.awt.event	java.awt.image
Adjustable(intf)	ActionListener(intf)	ImageConsumer(intf)
ItemSelectable(intf)	AdjustmentListener(intf)	ImageObserver(intf)
LayoutManager(intf)	ComponentListener(intf)	ImageProducer(intf)
LayoutManager2(intf)	ContainerListener(intf)	ColorModel
LightweightPeer (intf)	FocusListener(intf)	DirectColorModel
AWTError	ItemListener(intf)	IndexColorModel
AWTEvent	KeyListener(intf)	MemoryImageSource
AWTEventMulticaster	MouseListener(intf)	PixelGrabber
AWTException	MouseMotionListener(intf)	
BorderLayout	TextListener(intf)	
CardLayout	WindowListener(intf)	
Color	ActionEvent	
Component	AdjustmentEvent	
Container	ComponentAdapter	
Cursor	ContainerEvent	
Dimension	FocusAdapter	
Event	FocusEvent	
EventQueue	InputEvent	
FlowLayout	ItemEvent	
Font	KeyAdapter	
FontMetrics	KeyEvent	
Graphics	MouseAdapter	
GridLayout	MouseEvent	
IllegalComponentStateException	MouseMotionAdapter	
Image	PaintEvent	
Insets	TextEvent	
MediaTracker	WindowAdapter	
Panel	WindowEvent	
Point	ComponentEvent	
Polygon		
Rectangle		
Shape		
Toolkit		

The classes in Table 1 are not necessarily sufficient to enable a full implementation of a HAVi compliant device, for example a device implementing the HAVi User-Interface could be implemented using JDK 1.1, Personal Java , etc., which might require additional requirements on the implementation. The specification is intentionally silent on the mechanisms used to implement the Java environment for a HAVi implementation.

1.2.3 Additional Elements from AWT

Personal Java 1.1 includes some interfaces which are not found in JDK 1.1 but are useful for a TV friendly user-interface API. To provide the functionality offered by these interfaces without requiring Personal Java, the HAVi User-Interface includes the following synonymous HAVi interfaces, as detailed in Table 2.

Table 2. HAVi Interfaces and Synonymous Personal Java Versions.

HAVi interface	Personal Java AWT interface
org.havi.ui.HNoInputPreferred	com.sun.awt.NoInputPreferred
org.havi.ui.HKeyboardInputPreferred	com.sun.awt.KeyboardInputPreferred
org.havi.ui.HActionInputPreferred	com.sun.awt.ActionInputPreferred

These interfaces imply the same meaning as their corresponding Personal Java interfaces, whilst removing any implicit implementation requirements. The specification is intentionally silent on the mechanisms used to implement the Java environment for a HAVi implementation. An extra HAVi specific interface org.havi.ui.HAdjustmentInputPreferred is also included.

The org.havi.ui.HNoInputPreferred interface disallows user navigation, and hence actioning, etc.

A component that implements org.havi.ui.HNoInputPreferred indicates that the user may not navigate to this component. However, note that if a component which implements this interface is extended, so that the sub-classed component will implement another "XxxInputPreferred" interface, then in all cases, this other interface will take precedence. In contrast, the method isFocusTraversable should always return true for components implementing the interfaces org.havi.ui.HKeyboardInputPreferred, org.havi.ui.HActionInputPreferred and org.havi.ui.HAdjustmentInputPreferred.

The org.havi.ui.HKeyboardInputPreferred interface indicates that it is intended to accept alphanumeric input from the user. Platforms without keyboards will provide another means for generating such alphanumeric input when this component is edited, for example, by offering an on-screen keyboard.

The org.havi.ui.HActionInputPreferred interface indicates that it is intended to be actioned by the user.

The org.havi.ui.HAdjustmentInputPreferred interface indicates that it is intended to offer increment and decrement functionality to the user.

1.3 HAVi Extensions to AWT

1.3.1 User Input

Java Applications in HAVi can accept input from a keyboard, a mouse or a remote control. The keyboard and mouse inputs are supported by functions in the java.awt and java.awt.event packages. Remote control input is provided with classes in the org.havi.ui.event package. The org.havi.ui and org.havi.ui.event packages include classes that allow the application to determine the user-input capabilities of the platform on which the application is running.

1.3.1.1 Remote Control Support

The HAVi remote control classes are extended from the java.awt.event key event classes. All of the events that are added for the remote control are optional. The remote control keys fall into two categories: colored keys and dedicated keys. The intention of these keys is to provide the user direct access to various functions; however, the platform *may* implement a virtual (on-screen) mechanism to generate these events, but shall take care in this case not to hide the application. Note that it is an implementation option if (remote control) key events are repeated.

1.3.1.1.1 Remote Control Colored Keys

Up to six colored soft keys can be included on a remote control. These are optional, and are to be identified with a color. If implemented, these keys are to be oriented from left to right, or from top to bottom in ascending order. The application can determine how many colored keys are implemented, and what colors are to be used, so that the application can match the controls.

The following identifiers are available for colored key events: VK_COLORED_KEY_1, VK_COLORED_KEY_2, VK_COLORED_KEY_3, VK_COLORED_KEY_4, VK_COLORED_KEY_5, VK_COLORED_KEY_6.

1.3.1.1.2 Remote Control Dedicated Keys

The org.havi.ui.HRcEvent class defines a number of dedicated remote control events that can be used by applications. Although none of the events in the org.havi.ui.event.HRcEvent class are required to be implemented, events for power (VK_POWER), volume up and down (VK_VOLUME_UP and VK_VOLUME_DOWN), and channel up and down (VK_CHANNEL_UP and VK_CHANNEL_DOWN) are highly recommended.

However, whilst the dedicated remote control events are themselves device independent, the precise set of dedicated keys that is implemented is device dependent. The org.havi.ui.event.HRcCapabilities class enables an application to discover which events are implemented and how these are to be labeled to match the platform implementation.

1.3.1.2 Keyboard

HAVi supports keyboards via the java.awt.event package. The events supported on a keyboard can be determined by the org.havi.ui.event.HKeyCapabilities class.

Note that systems that do not include a physical keyboard can check each component to see if it implements org.havi.ui.HKeyboardInputPreferred. If this interface is implemented, the system shall enable user input of alphanumeric keyevents, for example, via a "soft" on-screen keyboard.

1.3.1.3 Mouse

Mouse support is optional. The presence of a mouse can be detected with the org.havi.ui.event.HMouseCapabilities class.

Mouse functionality is provided by the java.awt.event package. HAVi applications must be written in such a way that a free roaming cursor is not required for correct operation. This does not mean that a HAVi application could not implement, e.g. a drawing program, but rather that the user should not be able to put the application into a state that cannot be exited without a mouse. (A user-friendly drawing package would also notify the user that a mouse is required to use this application properly.)

1.3.1.4 User Input Capabilities

Three classes are available to determine the capabilities of the user input for a given platform: org.havi.ui.event.HKeyCapabilities, org.havi.ui.event.HMouseCapabilities, org.havi.ui.event.HRcCapabilities. Each of these classes includes a method called getInputDeviceSupported, which returns true if the particular device is known to be available.

1.3.1.5 User Input Representation

The org.havi.ui.event.HRcCapabilities class includes a method called getRepresentation, which returns an object of type org.havi.ui.event.HEventRepresentation. This class defines an event as having a known representation as a string, color or symbol, or having no supported representation. The particular text, color, or symbol can be determined by calling getString, getColor or getSymbol respectively. This allows an application to describe a button on an input device correctly for a given platform. All available events should have a text representation from getString.

The six colored key events (VK_COLORED_KEY_1 -- VK_COLORED_KEY_6), if implemented, must also be represented by a color – the getColor method returns a java.awt.Color object.

Key events may also be represented as a symbol – if the platform does not support a symbolic representation for a given event, then the application is responsible for rendering the symbol itself. Application rendering of keys without a symbolic representation, but with a commonly known representation, should follow the guidelines as defined in the javadoc definition of the class.

1.3.2 Graphics Devices and Configurations

1.3.2.1 Background

There are some specialized requirements for running applications within a consumer electronic environment, rather than the simpler situation that occurs when an Applet is displayed within a web-browser. Most notably the screen dimensions and aspect ratios are significantly different between PCs and CE devices. In the current on-screen display (OSD) graphics model of today's set-top box units, video may be output in a number of different configurations, e.g. traditional 4:3 TV display, or 16:9 widescreen TV displays, etc. The graphics resolution and aspect ratio are often locked to the video resolution and aspect ratio. If the video aspect pixel ratio changes then the graphics pixel aspect ratio may also change. Thus, there are requirements to:

- Determine the resolution and physical characteristics of the current display device.
- Detect modifications to the resolution and physical characteristics of the current display device.

1.3.2.2 The HAVi Screen Reference Model

HAVi provides a model for the video output from a consumer electronics device. Instances of the class HScreen represent each independent final video output signal from a device. Each independent final video output signal is made up from the sum of graphics devices, video devices and backgrounds. These are represented by instances of the classes HGraphicsDevice, HVideoDevice and HBackgroundDevice respectively. All of these classes inherit from a common parent class - HScreenDevice.

The HAVi User-Interface specification provides limited support for applications to be displayed so that they are split across multiple concurrent display devices – the HSceneFactory class allows the HGraphicsDevice to be specified in the HSceneTemplate used to generate the HScene's for the application.

1.3.2.3 The HAVi Screen Device Discovery Classes

HAVi defines a means to allow applications to discover the range of display devices available. The

model followed by HAVi is based on the model used in Java2 as described by the following three classes in the java.awt package - GraphicsDevice, GraphicsConfiguration and GraphicsConfigTemplate. In HAVi, this model is generalised to apply to video devices and to background devices.

1.3.2.3.1 Querying the Configuration of a Display Device

For each display device class (HVideoDevice, HGraphicsDevice and HBackgroundDevice), there are classes whose name ends in "Configuration" which represent distinct possible configurations of a single device. Applications may obtain a list of all possible configurations of a particular device. Applications may also obtain the current configuration using the getCurrentConfiguration method. Subject to security and resource management issues, applications may also set the configuration of a device using methods found on each device class.

Applications that are interested in a particular configuration of a device can request configurations matching a specific set of constraints. The first step in this process is to construct objects whose name ends in "ConfigTemplate". Instances of these classes can then be populated with the properties by the application and then used to request a configuration supporting those properties. Properties can also have priorities attached to allow applications to express whether support for that property is required by the application, whether support for that property is only preferred by that application, whether support for that property is required to be absent or whether support for that property is preferred to be absent. In some cases, properties such as PIXEL_ASPECT_RATIO require extra information. This extra information can be provided as part of the method used to add the property to the configuration template.

The Configuration for a Device can be acquired, using the getCurrentConfiguration method. A description of this Configuration can be obtained using the getConfigTemplate method that yields a ConfigTemplate that uniquely identifies the given Configuration. Individual properties in this ConfigTemplate can then be examined using the getPreferencePriority and getPreferenceObject methods – features that are implemented will return REQUIRED, features that are not implemented will return REQUIRED_NOT. Values of some properties may also be obtained through a limited set of query methods provided on HScreenConfiguration.

1.3.2.3.2 Compatibility with Existing java.awt Methods

The java.awt.Toolkit.getScreenSize method shall be equivalent to the pixel resolution of the current configuration of the default screen device returned by HScreen.getDefaultGraphicsDevice.

The java.awt.Toolkit.getScreenResolution method shall return the resolution (in dots per inch) of the current configuration of the default screen device returned by HScreen.getDefaultGraphicsDevice. In the case that the horizontal and vertical dpi resolutions differ, then the lower value shall be returned.

Where the screen aspect ratio is unknown (such as in the case where a set-top box is connected to an analog display), the default aspect ratio is 4:3. In the case where an analog monitor is used with a HAVi compliant set-top box the resolution returned shall be based on the raster of the set-top box, ignoring any interpolation or other processing that may be present in the monitor.

The java.awt.Toolkit.getNativeContainer method shall return null; interoperable applications should not rely on this method.

1.3.2.4 Detecting Configuration Changes on a Display Device

It is important for CE devices to be able to detect variations in their settings, since they may be subject to "on-the-fly" modifications of these settings, for example, they may be heavily influenced by the nature of some input video streams. Hence, the HScreenDevice class provides support for

detecting when its configuration (settings) have been changed, using the HScreenDevice.addScreenConfigurationListener methods and the HScreenConfigurationListener and HScreenConfigurationEvent classes.

When an HScreenDevice's configuration is modified, then an HScreenConfigurationEvent is generated. Note that after a HScreenConfigurationEvent is obtained any HScreenConfiguration (or HScreenConfigTemplate) associated with that HScreenDevice must be reacquired to obtain the current settings for the device.

In general, a modification to the HScreenDevice might require that the displayed user-interface be modified, e.g. if the resolution has changed, or the pixel aspect ratio has been modified.

1.3.2.5 Emulated Display Devices

The HAVi User-Interface introduces extra sub-classes that are used to indicate that a device may perform emulations of other device capabilities:

- HEmulatedGraphicsDevice
- HEmulatedGraphicsConfiguration

Instances of these classes can be returned by the same methods that would return the corresponding class without "Emulated" in the class name. Returning the sub-class indicates that the implementation is emulating the requested configuration on one of its actual supported configurations. The class HEmulatedGraphicsConfiguration includes methods to allow applications to compare the configuration being emulated and the actual underlying configuration being really used. The extent of support for emulated configurations is a profile issue. All possible emulated configurations are not required, or guaranteed to be supported. Emulated configurations may have a significant performance penalty with respect to those supported natively on the device.

1.3.2.5.1 Mapping from Authoring to Device Coordinates

A special case of device emulation is the emulation of various graphics coordinate systems on a single physical device. The HAVi User-Interface provides mechanisms that allow devices to perform such emulations, e.g. by down-sampling a high-resolution system to match the limitations of a standard definition display. Thus, authors can rely on seamless mapping between authoring and device coordinates by the use of the HEmulatedGraphicsDevice class. Authors may determine an appropriate graphics device and request the best configuration that matches their requirements, as in a standard device discovery mechanism – or examine configurations themselves (both emulation and implementation) to determine appropriate settings. The extent to which devices are required to support emulation of other coordinate systems is profile dependent.

1.3.2.6 Integrating HAVi Video Support into Platforms

The HAVi specification includes several classes to represent video in the user interface system. This representation of video devices only includes the display of video. The setup of the video decoder and the video pipeline is not included in this specification.

The class HVideoComponent is intended to be returned by a platform specific controller for video. In platforms based on the Java Media Framework, the Player.getVisualComponent method shall return objects of this class. The class HVideoDevice provides two hooks to platform specific APIs for this setup, the methods getVideoController and getVideoSource. On platforms based on the Java Media Framework (JMF), the getVideoController method shall return a JMF Player. The getVideoSource

method shall return a platform specific class encapsulating a reference to the source of the video. Possible examples of the class to be returned here could include java.net.URL or javax.media.MediaLocator.

1.3.2.7 Backgrounds

The HAVi specification includes several classes to represent the background of a screen, i.e. the area that is behind the running graphics and video and not covered by those. Using the same naming convention as video and graphics, these are called HBackgroundDevice, HBackgroundConfiguration and HBackgroundConfigTemplate. The basic HBackgroundConfiguration allows applications to control a single full screen background color.

The HAVi specification includes support for more sophisticated backgrounds - still images. Applications wishing to use these shall request an HBackgroundConfiguration supporting them by using the STILL_IMAGE property in an HBackgroundConfigTemplate. If this feature is supported by the platform concerned, when such a configuration is requested, an instance of the class HStillImageBackgroundConfiguration shall be returned. This class adds two extra methods, over the standard background configuration, which support the loading of background images. This loading is done through the HBackgroundImage class.

Using the HBackgroundImage class rather than the standard java.awt.Image allows for image formats that are decoded using hardware outside of the graphics system. One specific example of this is the decoding of still MPEG I frames using an MPEG video decoder. This is a commonly used feature in some devices since it provides good quality backgrounds without using software decoders or system memory. In systems where the same underlying MPEG video decoder can be used to decode both video and MPEG I frames, this decoder shall be represented both by an HVideoDevice instance and by an HBackgroundDevice instance for each application. The sharing of the single underlying decoder between these two instances shall be managed using the reserveDevice and releaseDevice methods that both classes inherit from HScreenDevice. This applies both for resource transfers between two different applications and for resource transfers within the same application. The behavior of this shall be as described in the specification for ResourceClient.

For example, on such a system, in the situation where an application has the right to control an HVideoDevice and that same application wishes to request control of a HBackgroundDevice currently configured as an HStillImageBackgroundConfiguration, the following steps are followed. First, the application calls reserveDevice on the HBackgroundDevice. This method will block while the following steps complete. The implementation calls the method ResourceClient.requestRelease on the ResourceClient provided by the application when it called HVideoDevice.reserveDevice. The implementation of that method shall decide whether to release the underlying decoder for use in decoding backgrounds or to continue with decoding video. If the application decides that the use of the background is more important to it than the use of the video then the application shall conform to the behavior defined for this method, i.e. call HVideoDevice.releaseDevice and then return true. At this point, the original call to reserveDevice shall return true. On systems where there are separate underlying decoders for MPEG video and MPEG stills, clearly calling HBackgroundDevice.reserveDevice will return immediately if no other applications are using that device.

1.3.2.8 Control of Screen Configurations

The HAVi specification is silent about whether a single display device is shared between multiple applications or not. For the case where a display device may be shared between applications, it provides a mechanism for applications to assert control over the right to change the configuration of the display device. The HScreenDevice class includes methods to allow applications to reserve and release the right to control this configuration. It also allows an application to register and remove

listeners for events that are generated when the applications reserve and release this right.

Applications wishing to be able to control the configuration of an HScreenDevice must define a class implementing the ResourceClient interface and pass an instance of this class to the reserveDevice method of the HScreenDevice that they wish to control. If the reserveDevice method succeeds then the application obtains control over the device configuration. When an application calls the HScreenDevice.getClient method this will return the ResourceClient passed in to the last call to the reserve method on that HScreenDevice instance.

Where there is a conflict between applications, this specification includes a mechanism to allow the platform to arbitrate between conflicting applications. The policy for this arbitration is intentionally not defined in this specification. When it is decided to remove the right to control a screen from an application, this is notified through the ResourceClient interface, the notifyRelease method will always be called. The requestRelease method will only be called when the existing owner of the resource and the application requesting the resource are authenticated to have a secure relationship of some form. This specification is silent about the details of this authentication.

1.3.3 Graphics and Video Integration

1.3.3.1 Configurations

The HAVi specification allows applications to express the relationship between video, graphics and backgrounds. The method HScreen.getCoherentScreenConfigurations allows applications to express a common set of constraints for video, graphics and backgrounds and get back a coherent answer.

In addition to this, there are several means to express constraints between video and graphics. These can be used for applications which already have running video to fit a graphics configuration to that video or which have already running graphics to fit video to that graphics. 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_REGISTERED - video & graphics pixels are the same size and aligned
- ZERO_VIDEO_IMPACT - a new graphics configuration must not change the existing video configuration
- ZERO_GRAPHICS_IMPACT - a new video configuration must not change the existing graphics configuration

1.3.3.2 Coordinate Spaces

The HAVi specification includes a normalised screen coordinate system that represents the coordinates on the screen as floating point numbers between zero and one. This coordinate system is not pixel based. Such a non-pixel-based coordinate system enables the following:

- meaningful results, even when the graphics configuration has not been determined
- meaningful results when presented video does not have a java.awt component.

- meaningful results when the video display and the graphics display are not necessarily aligned / share the same origin / share the same resolution, etc.

This screen-based coordinate system is encapsulated in the HScreenRectangle and HScreenPoint classes. This specification is silent about conversion between normalised and video coordinates. This should be addressed as part of the API providing support for control of video.

For graphics, these conversion mechanisms are found on the HGraphicsConfiguration class, since the conversion from screen to graphics coordinates is dependent on the current graphics device settings (Configuration) – especially if e.g. the graphics resolution can be varied independently of the video resolution, etc.

The HScreenRectangle mechanisms can be used to enable the alignment of (portions of) video and (portions of) graphics. The HScreenLocationModifiedListener and HScreenLocationModifiedEvent allow mechanisms to determine if the on-screen location of an HVideoComponent is modified (rather than its relative location within its enclosing container).

1.3.3.3 Transparency between Graphics and Video

The HAVi specification includes support for applications to request transparency between graphics and video. This is provided by the getPunchThroughToBackgroundColor method on the HGraphicsConfiguration class. These methods provide a factory that enables applications to provide an opaque java.awt.Color and obtain a java.awt.Color supporting some form of transparency between graphics and video. These Color objects may be used in the drawing methods in the java.awt.Graphics class to cause video to appear in the graphics system.

1.3.4 HSceneFactory, HSceneTemplate and HScene

The HAVi User-Interface is deliberately agnostic concerning the implementation of a "coordinating" environment that provides the mechanism for a user to choose and run one or more applications. In HAVi, this "coordinating" environment is known as a "home navigation shell". Application writers cannot make any assumptions that their application GUI will always be immediately visible. For example, valid implementations of a coordinating environment might include:

- A simple "full-screen" view on a single application at any one time (with some undefined mechanism to switch between them).
- A multi-window system, where windows may obscure each other.
- A "paned" system where each application occupies an area on-screen – i.e. each application is always visible, but they may be resized if other applications are installed.

Thus, mechanisms are required to initiate an on-screen display and to indicate "user-interest", or other modifications to the rendering area. These mechanisms should:

- Enable an application to request an area on-screen – however, given the possibility of differing styles of coordinating environment, an application cannot reasonably expect that its request will always be honoured perfectly, and thus, a mechanism is required to indicate preferences for the application location "on-screen".

- Indicate whether the current application is the one which the user is specifically interested in. For example, a "well-behaved" application which the user is not currently using might release, or reduce its consumption of any limited resources.

- Indicate to an application that its extent and position on-screen have been modified somehow by the home navigation shell.

- Allow an application to indicate to the system, that it requires the user's attention. For example, the system may either indicate to the user that the user should choose the indicating application, or might simply automatically switch to that application.

1.3.4.1 Requesting an Area On-screen

1.3.4.1.1 HSceneFactory and HSceneTemplate

The HSceneFactory is a factory class that is used to generate HScene objects. An application can indicate the location and dimensions of the HScene in the associated HSceneTemplate, although it is not guaranteed that the resulting HScene will necessarily match all of these preferences – since this is dependent on the implementation of the controlling shell and its associated policies, etc.

The application should call HSceneFactory.resizeScene if it wishes to re-size the HScene.

1.3.4.1.2 HScene

An HScene is an HContainer representing the displayable area on-screen within which the application can display itself and thus interact with the user. However, HScene does not paint itself on-screen, only its added "child" components and hence there is no requirement to allocate "pixels" to the HScene directly – its only effect is to "clip" its child components. Hence, HScene may be regarded as a simple connection to the window management policy within the device, acting as a "screen resource reservation mechanism" denoting the area within which an application may wish to present a component, at some point in the future. Since an HScene is by definition not painted, ie it is effectively transparent, the area behind (all) HScene's in the z-ordering may be exposed by the platform as an HBackgroundDevice, and/or HVideoDevice's. However, HAVi does not require platforms to provide such device capabilities, this is platform specific. The HScene semantics for transparency need to be specified exactly on a per-platform basis, for example, on some platforms an HScene might be transparent to other HScene's due to other separate applications.

For all interoperable applications, the HScene is considered the main top-level component of the application. No parent component to an HScene should be accessible to applications. Interoperable applications should not use the getParent method in HScene, since results are implementation dependent and valid implementations may generate a run-time error.

In terms of delegation, the HScene shall behave like a java.awt.Window with a native peer implementation, in that it will not appear to delegate any functionality to any parent object. Components which do not specify default characteristics inherit default values transitively from their parent objects. Therefore, the implementation of HScene must have valid defaults defined for all characteristics, e.g. Font, foreground Color, background Color, ColorModel, Cursor and Locale.

The HScene has a null LayoutManager by default – all widgets are placed using an X, Y co-ordinate, specified by the widget.

When created an HScene is not initially visible, and a call to setVisible is required to display the HScene (and also to hide it).

The application should call HSceneFactory.dispose if it wishes to destroy the HScene (and all of its currently added Components) and therefore release their associated resources for future garbage collection by the platform. After calling this method, any further method calls on the HScene will result in a java.lang.IllegalStateException being thrown.

1.3.4.2 Modifications to the HScene: Focus and Resize events

The HScene object accepts java.awt.event.WindowEvent's, and interprets them as a java.awt.Window, however it is not required for the home navigation shell to generate all types of java.awt.event.WindowEvent.

Applications can use the java.awt.Component.requestFocus method on the HScene to indicate to the home navigation shell that the HScene should be receiving input focus. This request should be treated as a request to make the entire application visible and ready for user input, e.g. by expanding an icon, or changing the stacking order between competing overlapping applications. The decision as to whether or whenever the HScene (application) gains the input focus is entirely platform specific in terms of policy, etc. The java.awt.Component must be visible on the screen for this request to be granted – note that visibility in this context refers to whether the application has called the HScene.setVisible method, rather than any possible non-application-defined-behavior, due to the action of the coordinating shell hiding an application.

The java.awt.event.ComponentEvent's COMPONENT_MOVED and COMPONENT_RESIZE will be received by the HScene when the controlling shell has modified the position of the HScene or changed its dimensions on screen, respectively.

1.3.4.3 Application "user-interface" Lifecycle

- Outside the scope of the HAVi User-Interface:
  - The application is acquired by the platform.
  - The application is validated and security checked (possibly including authentication, byte-code verification, etc.).
  - The virtual machine is initialised, ClassLoader created, etc.
  - The application is executed
  - If the application does not require a user-interface, then it may continue as per normal.
- If a user-interface, and hence some screen resource is required, then the application traverses the HScreen, HGraphicsDevice, HGraphicsConfiguration space to determine an appropriate configuration, using HGraphicsConfigTemplate e.g. video-mixable, full-screen graphics, square pixel aspect ratio, resolution 1280 by 1024.
- The application configures the HGraphicsDevice appropriately, using the setGraphicsConfiguration method.
- The application requests that the HSceneFactory effectively grant it access to part of the screen for that device, using HSceneTemplate, e.g. full-screen display.
- The HSceneFactory returns an appropriate HScene container within which the application can display itself.
- The application uses the HScene container to add all of its components to make its user-interface.

- The application may take advantage of java.awt.WindowEvent's, to determine whether it has the user's (input) focus.
- The application may take advantage of the events COMPONENT_RESIZED and COMPONENT_MOVED, to determine when its HScene extent / location has been modified and to tailor its presentation accordingly.
- The application resizes the HScene by using HSceneFactory.resizeScene – if it wishes to resize the HScene, with the caveat that this may not be allowed by the external environment, e.g. due to window-manager policy, etc.
- The application terminates its on-screen presentation by calling the HScene.dispose method
- Outside of the scope of the HAVi User-Interface:
  - The application itself terminates.

1.3.5 Effects and Visual Composition using Component Mattes

1.3.5.1 Component Mattes

With org.havi.ui, the user interface is constructed from a set of components arranged in a hierarchy. The root of the hierarchy is an instance of HScene, leaf nodes are instances of HComponent and intermediate nodes are instances of HContainer. Components within a container are ordered from back to front. An example is shown in Figure 1, where *c3*, a container, is the back most component and *c1* the front most.

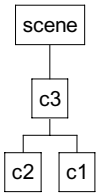


Figure 1. Scene Hierarchy

With the HMatte interface, the scene hierarchy can be modified by the inclusion of mattes (additional alpha sources), potentially for each member, i.e.:

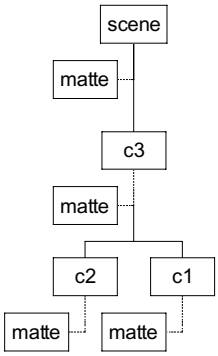


Figure 2. Scene Hierarchy with Mattes

The mattes influence the rendering of the scene, their operation can be visualised using a 2_D or layering model. The example below corresponds to the hierarchy in Figure 2 (for simplicity, the scene matte is not shown).

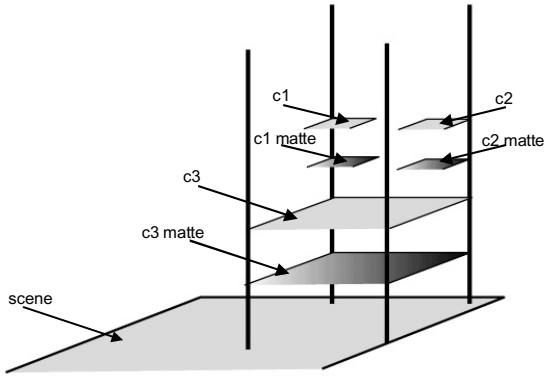


Figure 3. Component Mattes

Where pixels in a component already have an alpha value (e.g., from a PNG image), the alpha value from the component and the alpha value from the matte are multiplied together to obtain the actual alpha value to be used for that pixel.

1.3.5.2 Component Grouping

A container is either “grouped” or “ungrouped”. When a container is ungrouped, its matte only influences the appearance of those regions of the container not covered by members of the container (i.e., exposed regions of the container’s background). When a container is grouped, its matte influences the appearance of its background and all members of the container. For example, grouping a container and setting its matte to indicate 50% transparency will fade the container’s background and all members of the container. If it is ungrouped only the background will fade.

An HContainer may be rendered as follows:

- If the container is ungrouped, the container's background is first rendered and then composited with the container's matte (i.e., the RGBA value of the container's background is combined with the alpha value from the matte). Then, in back to front order, each member of the container is rendered, composited with its matte, and then composited with the container.

- If the container is grouped, the container's background is first rendered. Then, in back to front order, each member of the container is rendered, composited with its matte, and then composited with the container. The result is then composited with the container's matte.

After an HContainer is rendered, it is composited with its parent. Compositing of an HScene is determined by the configuration of display devices.

1.3.5.3 Examples of Mattes and Component Composition

a) Bar matte for lower component.

b) As in a), with top components grouped to lower.

c) Circular mattes for top components.

d) As in c), with bar matte for lower component.

e) As in d), with top components grouped to lower.

Figure 4. Visual Composition Examples

1.3.5.4 Effects

A great variety of effects (e.g., wipes and fades) can be performed by using *matte animations* –

sequences of mattes where the "active" element is changed over time. Matte animations can be combined with other techniques, such as component movement, to produce additional effects. The construction of matte animations is facilitated by the following classification of mattes:

HFlatMatte – the matte is constant over space and time, it can be specified by a float (0.0 is fully transparent and 1.0 fully opaque)

HImageMatte – the matte varies over space but is constant over time, it can be specified by an "image mask" (a single channel image) where the pixels indicate matte transparency

HFlatEffectMatte – the matte is constant over space but varies over time, it can be specified by a sequence of floats

HImageEffectMatte – the matte varies over space and time, it can be specified by a sequence of image masks

1.3.5.5 Matte Sizes and Offsets

When a HImageMatte or HImageEffectMatte is assigned to a component, the associated image (or images) is by default aligned with the component so that their origins – the pixel at (0,0) – coincide. The offset of the matte with respect to the component can be altered using the setOffset method of HImageMatte and HImageEffectMatte. Regions of the component outside the matte (resulting from either a matte being smaller than the component, or from shifting the matte) are not matted.

1.4 HAVi Widget Framework

The HAVi widget framework is designed to allow maximum flexibility to implementers of applications. It also provides the necessary extensibility to allow the widget framework to be used as the basis for other application types, such as broadcast applications. By default, the HAVi widget framework only copies object references, and does not clone objects. Cases where objects are clone'd shall be marked explicitly.

1.4.1 HAVi Event Mechanism

The HAVi event mechanism is intended to mirror that used in the AWT. HAVi User-Interface events are based on java.awt.event.KeyEvent. The key codes are enumerated in the HUIEvent class, and are shown in the table below.

Table 3. HUIEvent Codes

Keycode	Meaning
VK_ACTION	HActionable widget should perform action
VK_START_CHANGE	HValue widget should enter change mode
VK_END_CHANGE	HValue widget should end change mode
VK_NEXT_CHAR	Text entry widget should move caret forward
VK_PREV_CHAR	Text entry widget should move caret back
VK_NEXT_LINE	Multiline text entry widget should move caret down
VK_PREV_LINE	Multiline text entry widget should move caret up
VK_ADJUST_MORE	HValue widget should increase value
VK_ADJUST_LESS	HValue widget should decrease value

The intention of these events is to provide a means for HAVi widgets to receive events in a platform independent manner without impairing performance.

A HAVi widget must respond to HAVi UI key events in addition to other applicable user-input mechanisms. However, widgets are not required to consider all of the HAVi User-Interface events, for example, caret navigation key events are only meaningful to text entry widgets (that are already in their editing mode). This mechanism allows platforms that do not physically have separate keys that generate these events to generate the events from other keys, e.g. if separate caret positioning keys are unavailable then the navigation keys could be "re-used" for editing.

1.4.2 Abstraction of "Feel"

In order to provide the necessary flexibility, the HAVi User-Interface widget framework is defined around a core of abstract *Component Behaviors*. These effectively define the functionality (or "feel") of each widget which is derived from one of the Component Behaviors. Behaviors are defined for widget types having a number of states, which may be used to mimic the behavior of typical widgets.

In summary these Component Behaviors are:

- HVisible – a simple, one-state Behavior providing basic display functionality.
- HNavigable– a two-state Behavior enabling widgets to receive navigational focus, and to define some kind of display change associated with focus change.
- HActionable– a three-state Behavior providing a third "actioned" state and allowing functionality to be invoked in response to that action.
- HSwitchable– a four-state Behavior allowing a widget to be actioned and to retain internal state information in addition to simple action behavior.
- HValue– a Behavior permitting the definition of widgets that return values to applications in response to user interaction.

Based upon these fundamental abstract Behaviors, all necessary HAVi functionality can be provided through derived concrete widgets, either for the provision of HAVi specific user-interfaces, or for HAVi specific widgets. In addition, these abstractions form the basis upon which other interactive applications may be built without the requirement for the use of HAVi specific widgets. Thus, the HAVi widget framework is more generally applicable to interactive application

execution, rather than exclusively focused upon HAVi.

1.4.3 Framework Class Hierarchy

The HAVi widget framework consists of a base class (HVisible) and a set of interfaces that model the behaviors different types of widget may exhibit. The behavior is modeled on the number of states a widget may represent. For each such state a widget can present a particular representation (graphical, textual and sound) to the user.

The widget framework allows for simple user interface development by application authors. It also reduces the size of the developed application, since most of the presentation and interaction capability is resident on the device – developers can concentrate on the specific functionality of their application.

1.4.3.1 HContainer

Components in the HAVi User-Interface are explicitly allowed to overlap each other. Hence, the HAVi User-Interface extensions adds additional Z-ordering related methods to org.havi.ui.HContainer:

Additional semantics related to transparency of the HContainer itself and its Components, are also defined via the HMatteLayer interface.

The org.havi.ui.HContainer class also adds the ability to determine whether hardware double buffering is present, using the isDoubleBuffered method.

The org.havi.ui.HContainer class also adds the ability to determine whether it is completely opaque, by applications overriding the isOpaque method.

Additionally, the default LayoutManager for HContainer is defined to be null, i.e. absolute positioning, in contrast to the FlowLayout used in java.awt.Container.

1.4.3.2 HComponent

The base class for all HAVi widgets.

The org.havi.ui.HComponent class extends java.awt.Component to include additional semantics related to transparency of the HComponent, defined via the HMatteLayer interface.

The org.havi.ui.HComponent class also adds the ability to determine whether hardware double buffering is present, using the isDoubleBuffered method.

The org.havi.ui.HComponent class also adds the ability to determine whether it is completely opaque, by applications overriding the isOpaque method.

1.4.3.3 HVisible

Represents a widget that has only a single state, for example HStaticText or HIcon.

1.4.3.4 HNavigable

An interface that is implemented by classes that are derived from HVisible for adding an additional

state that is used to indicate if the widget is currently focused.

The HNavigable interface also provides the functionality necessary to manage the focus navigation between widgets assuming a remote control style UP, DOWN, LEFT, RIGHT form of navigation, using the setFocusTraversal method.

The precise semantics of the HNavigable interface are defined in the supporting javadoc.

1.4.3.5 HActionable

The HActionable interface extends HNavigable by adding an additional state that is used to indicate when the widget has been actioned.

The HActionable interface provides the functionality necessary to associate ActionListeners with the widget, using the addActionListener and removeActionListener methods. These ActionListeners will be called when the widget is actioned.

A widget that implements the HActionable interface is actioned when it receives a havi.ui.event.HUIEvent.VK_ACTION key event. The widget will move into its Actioned state by presenting its Actioned look. Any associated ActionListeners will be called by the widget calling its HActionable.processActionEvent method. When the ActionListeners have returned the widget will return to its focused state.

The precise semantics of the HActionable interface are defined in the supporting javadoc.

1.4.3.6 HSwitchable

The HSwitchable interface extends HActionable by adding an additional state that is used to maintain an internal (on/off) value.

The state transitions for HSwitchable are as follows:

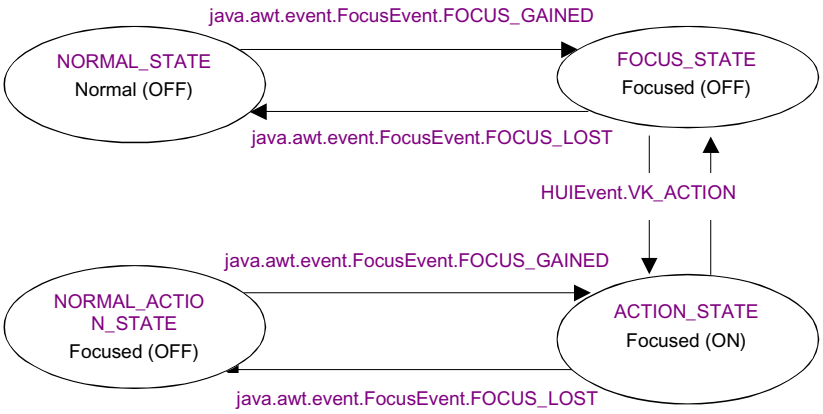


Figure 5. HSwitchable Transitions

The precise semantics of the HSwitchable interface are defined in the supporting javadoc.

1.4.3.7 HValue

The HValue interface extends HNavigable by adding support for managing a widget with an internal value that can be manipulated by user interaction.

The HValue interface provides the functionality necessary to associate HValueChangeListeners with the widget, using the addChangeListener and removeChangeListener methods. A widget that implements the HValue interface generates HValueChangeEvents when the value of the widget is initially changed, when further modifications to the value are made and when the final value is set. HValueChangeListeners can be associated with the HValue widget to process these HValueChangeEvents.

The precise semantics of the HValue interface are defined in the supporting javadoc.

1.4.4 Separation of "Look"

The flexibility of the HAVi widget framework is further enhanced by separating the "look" component from that for "feel". This allows easy construction of many styles of presentation associated with each of the abstract Component Behaviors defined previously.

Content can be associated with each state of a widget. For each widget state, textual, graphical and user defined content can be associated with the widget. The HLook interface defines the mechanism by which the content for the particular state of the widget can be rendered.

The HLook method showLook is used to provide the rendering of the content for the widget. This method will be called in response to a paint method call of the associated lightweight component. showLook is similar to the lightweight component's paint method, however because this method is separated from the widget class, there is no need to subclass the widget to change its look. The showLook method is responsible for repainting the entire component, including its background, subject to the clipRect of the Graphics object passed to it. The showLook method should not modify the clipRect of the Graphics object that is passed to it.

An HLook may also provide some form of border decoration, for example, drawing a rectangle around the widget when it has focus. To allow for predictable layout and presentation the HLook interface provides methods that are used to indicate the size of such a border area.

To support layout managers the HLook interface also defines the following methods, which allow the associated HVisible to query the HLook for its maximum, minimum and preferred sizes: getMaximumSize, getMinimumSize, getPreferredSize.

1.4.5 Pluggable Looks

The HAVi Widget framework provides a set of standard classes that implement the HLook interface. These can be regarded as the set of default looks that will be provided by all implementations. The particular rendering of a look is not defined and is manufacturer dependent.

Pluggable Look is defined in such a way as to allow implementers to extend the number of "looks" available to their application. By doing so, new "looks" are automatically available for every Component Behavior and for all widget types derived from those Behaviors. This is described in the Pluggable Look Interface.

To facilitate application development and to limit the size of applications, a set of pre-defined "looks" is provided. These are:

- HAnimateLook – presentation of an animated image sequence.
- HGraphicLook – presentation of graphical content.
- HRangeLook – presentation of a value within a range.
- HTextLook – a simple presentation mechanism for textual content.
- HSinglelineEntryLook – presentation of a single line of textual content that can be edited by the user.
- HMultilineEntryLook – presentation of multiple lines of textual content that can be edited by the user.

This basic set allows the construction of most typical interactive user interfaces when used in conjunction with the Component Behaviors to define a widget set. It can however be extended in a general fashion to provide new categories of "look".

When a widget is constructed, it is provided with a default look. This default HLook will be one of the standard set of looks listed above. For example, the HGraphicButton is created with the HGraphicLook by default. The default look that is used when the widget is constructed can be changed by calling the static method setDefaultLook that is provided on all widget types. Any widget of that type created after the call will be created with the new HLook that was passed in as the parameter to setDefaultLook. The look of an individual widget can be modified by using the method HVisible.setLook.

The Pluggable Look mechanism is flexible enough so that the application developer can create new HLooks. For example a combined HGraphicLook and HTextLook, where the Text may overlay the Graphic, or be shown in place of the Graphic while the Graphic is being loaded.

1.4.6 Content Behavior

Content is associated with the widget through the following methods on HVisible: setTextContent, setGraphicContent, setAnimateContent and setContent. Hence, multiple content (Text, Graphics, Animations and user-defined content) can be associated with a widget. The way multiple content is rendered is dependent on the HLook associated with the widget. The default looks provided by the platform may not render all the content types.

Different content can be associated with the different states of the widget. For example, an HGraphicButton might have three different images to represent its three different states to the user, using the setGraphicContent. The same content can be applied to all states of the widget by using the HState constant ALL_STATES when calling setTextContent, setGraphicContent, setAnimateContent and setContent.

By default, content associated with a widget is not modified to fit the dimensions of the widget, and thus the default behavior for the predefined looks is as follows:

- If the widget size is smaller than the content, the content will be drawn from the top left hand corner of the widget and cropped to the size of the widget.
- If the widget is larger than the content, the widget will be filled with the background colour of the component. The content will then be drawn from the top left hand corner of the widget.

Mechanisms are also available that allow (graphic) content to be resized to match the widget dimensions, etc.

1.5 HAVi Resident Widgets

Using the Component Behaviors (HVisible, HNavigable, HActionable and HSwitchable) defined in the previous section a set of resident widgets is provided.

Note that implementations of the HAVi widget set shall be implemented (and behave) as lightweight components. HAVi widgets do not include an associated peer class, irrespective of the exact mechanism for their implementation, i.e., directly implemented in Java, or via some platform specific mechanism.

1.5.1 Simple Text/Graphic/Animate Widgets

The HAVi set of resident widgets includes both visible and navigable versions of the HText, HIcon and HAnimation classes:

- Applications providing simple "display-only" non-navigable text, image, or animations may employ the "Static" versions of these classes.
- Applications wishing to provide additional feedback, e.g. "tooltips", or audio feedback – for example, a commentary – may employ the navigable versions of these classes.

Widget Type	Description	Static	Navigable
Animation	Displays a simple sequence of images	HStaticAnimation	HAnimation
Text	Displays a text label	HStaticText	HText
Graphic	Displays an Image	HStaticIcon	HIcon

Refer to the supporting javadoc for a more detailed description of these widgets.

1.5.2 Buttons

The HAVi set of resident widgets includes both textual and graphical version of a push button: HTextButton and HGraphicButton. These buttons implement the HActionable interface that defines their behavior.

The HToggleButton is used to represent a graphical control that has a boolean state that can be toggled on and off by the user (e.g. Checkbox or Radio Button). The HToggleButton implements the HSwitchable interface that defines its behavior. A HToggleButton widget does not have an associated text label as part of the widget. If a text label is required, a separate HStaticText widget should be created.

A set of HToggleButtons can be associated with a HToggleGroup. A HToggleGroup will ensure that a maximum of one HToggleButton is chosen at any time (i.e. a group of radio buttons).

Refer to the supporting javadoc for a more detailed description of these widgets.

1.5.3 Range Widgets

The HAVi set of resident widgets include a group of controls to represent a particular integer value in a range of values i.e. a slider control, or scroll bar. The HStaticRange widget is a non navigable widget, HRange widget is navigable (implements the HNavigable interface) and the HRangeValue is navigable and its value can be modified by user interaction (implements the HValue interface).

Refer to the supporting javadoc for a more detailed description of these widgets.

1.5.4 List Widgets

A HListGroup is an HContainer that manages a dynamic set of vertically or horizontally scrollable HListElements, allowing either single or multiple HListElements to be chosen by the user. The HListGroup will automatically scroll the HListElements when the user navigates to an element that is currently not visible within the list group.

Refer to the supporting javadoc for a more detailed description of these widgets.

1.5.5 Text Entry Widgets

The HSinglelineEntry component allows a user to enter a single line text string. A typical rendering is as a text entry field, e.g. with an associated on-screen keyboard. The HMultilineEntry widget extends the HSinglelineEntry widget and allows text to be entered over multiple lines. Both these widgets implement the HValue interface resulting in the widgets firing HValueChangeEvents when starting to edit, finishing editing, and whenever the content changes.

Refer to the supporting javadoc for a more detailed description of these widgets.

1.5.6 HDialog

The HDialog class extends the HContainer class to allow a modal dialog/pop-up to be displayed by the application. In an enhanced broadcast environment, it is likely that many applications will wish to inform, or confirm information with the user (e.g., confirm to the user that the modem is to be used). The HDialog is used by adding Components to it, and then calling startDialog. A button will usually be added that has an ActionListener that calls endDialog e.g. the OK and Cancel buttons.

To allow maximum flexibility in implementation, no parent component to an HDialog should be accessible to applications. Interoperable applications should not use the getParent method in HDialog, since results are implementation dependent and valid implementations may generate a run-time error.

In terms of delegation, the HDialog shall behave such that it has valid defaults defined for all characteristics, e.g. Font, foreground Color, background Color, ColorModel, Cursor and Locale, which should be equivalent to those for its corresponding HScene.

If a background image is specified for the HDialog, it is assumed that this image will also provide the appearance of the frame bounding the HDialog, and therefore the HDialog will not draw a rectangle around itself.

An HDialog is invisible until the application calls startDialog. This method will block until a call to endDialog.

An application can indicate the importance of the HDialog by passing an urgency flag through the

startDialog method. If the urgency flag is set the dialog will request from the home navigation shell that the application and hence dialog are visible and have input focus.

To support message box type functionality where a simple “Yes”/ “No” or “OK”/ “Cancel” request is required from the user the endDialog method can return a value back to the main application through startDialog.

The precise semantics of the HDialog are defined in the supporting javadoc.

Package org.havi.ui

Class Summary

Interfaces

HActionable	This interface is implemented for all user interface components that are actionable (ie have three states: NORMAL_STATE , FOCUS_STATE and ACTION_STATE).
HActionInputPreferred	A component which implements HActionInputPreferred indicates that it is intended to be actioned.
HAdjustmentInputPreferred	A java.awt.Component which implements HAdjustmentInputPreferred indicates that this component does not expect to receive all subclasses of java.awt.event.KeyEvent, but simply expects the HUIEvent adjustment input events VK_ADJUST_MORE and VK_ADJUST_LESS .
HAnimateEffect	The HAnimateEffect interface defines effect constants and controls for time-varying animations.
HKeyboardInputPreferred	A component which implements HKeyboardInputPreferred indicates that alphanumeric entry is required.
HLook	The HLook interface defines the "look" of a component and may be regarded as a mechanism to allow a "pluggable" paint method to be attached to the component.
HMatte	HMatte is the base interface for all matte classes.
HMatteLayer	This HMatteLayer interface enables the presentation of components, together with an associated HMatte , for matte compositing.
HNavigable	The HNavigable interface is implemented by HAVi UI components that can be navigated to by the user (i.e.
HNoInputPreferred	A component which implements HNoInputPreferred indicates that the user cannot navigate to this component.
HState	The HState interface encapsulates constants for widget states which are used in the HVisible setContent and getContent methods, to indicate which state the specified content is to be set.
HSwitchable	The HSwitchable interface is implemented for all user-interface components that can be toggled on or off (ie have four states, NORMAL_STATE , FOCUS_STATE , ACTION_STATE and NORMAL_ACTIONED_STATE)
HTextLayoutManager	The HTextLayoutManager class manages the layout and rendering on-screen of a "marked-up" string.
HValue	This interface is implemented for all widget types that represent some form of value (eg a range control or text edit).
HVersion	The HVersion interface defines some versioning constants that are accessible by using the java.lang.System method getProperty, with the appropriate property name.

Classes

HAnimateLook	This class defines the HAnimateLook of an HVisible and is the default HLook that is used by HStaticAnimation and its subclasses.

Class Summary

HAnimation	The HAnimation class is a user interface component used to display a sequence of images (as HStaticAnimation) which additionally enables a user to navigate (focus) upon it.
HBackgroundConfigTemplate	The HBackgroundConfigTemplate class is used to obtain a valid HBackgroundConfiguration .
HBackgroundConfiguration	The HBackgroundConfiguration class describes the characteristics (settings) of an HBackgroundDevice .
HBackgroundDevice	This class represents the ultimate background of a screen.
HBackgroundImage	This class represents a background image.
HComponent	The HComponent class extends the java.awt.Component class by implementing the HMatteLayer interface.
HContainer	The HContainer class extends the java.awt.Container class by implementing the HMatteLayer interface and providing additional z-ordering capabilities, which are required since components in the HAVi user-interface are explicitly allowed to overlap each other.
HDefaultTextLayoutManager	The HDefaultTextLayoutManager provides the default text rendering mechanism for the HStaticText HText and HTextButton classes.
HDialog	The HDialog class extends the HContainer class to allow a modal dialog/pop-up to be displayed by the application.
HEmulatedGraphicsConfiguration	A HEmulatedGraphicsConfiguration is a configuration for a "virtual" graphics device that may perform one or more emulations, e.g.
HEmulatedGraphicsDevice	A HEmulatedGraphicsDevice is a "virtual" graphics device that has the capability to be configured to perform one (of many) possible emulations.
HFlatEffectMatte	The HFlatEffectMatte class represents a matte that is constant over space but varies over time, it is specified as a sequence of floats.
HFlatMatte	The HFlatMatte class represents a matte that is constant over space and time, it is specified as a float (0.0 is fully transparent and 1.0 fully opaque).
HFontCapabilities	The HFontCapabilities class allows applications to query the rendering support for various character ranges / individual characters within specified fonts.
HGraphicButton	The HGraphicButton class is used to represent a conventional push-release button used for user actions.
HGraphicLook	The HGraphicLook class is used to getGraphicContent for the current state of the associated HVisible to determine the content to render, and render this on- screen to represent the HVisible .
HGraphicsConfigTemplate	The HGraphicsConfigTemplate class is used to obtain a valid HGraphicsConfiguration .
HGraphicsConfiguration	The HGraphicsConfiguration class describes the characteristics (settings) of an HGraphicsDevice .
HGraphicsDevice	The HGraphicsDevice class describes the raster graphics devices that are available for a particular HScreen.
HIcon	The HIcon class creates an a graphical image.
HImageEffectMatte	The HImageEffectMatte class represents a matte that varies over both space and time, it is specified as a sequence of image masks.
HImageHints	The HImageHints object allows an application to pass hints to the system how best to tailor an image to match a (possibly) restricted HGraphicsConfiguration .

Class Summary	
HImageMatte	The HImageMatte class represents a matte that varies over space but is constant over time, it can be specified by an "image mask" (a single channel image) where the pixels indicate matte transparency.
HListElement	The HListElement class is an HSwitchable component and hence maintains a switchable state as to whether it is currently "selected" or not, as would be expected in a list type control.
HListGroup	A HListGroup manages a set of HListElement and presents these elements to represent a scrollable list.
HListGroupLayoutManager	A HListGroupLayoutManager arranges HListElement in a vertical or horizontal list within the HListGroup container.
HMultilineEntry	The HMultilineEntry class is used to receive multiple lines of alphanumeric entry from the user.
HMultilineEntryLook	The HMultilineEntryLook class is used by the HMultilineEntry component to display the entering of text.
HRange	The HRange component is used for displaying a value which is within a fixed range.
HRangeLook	The HRangeLook class displays a slider type range control on screen.
HRangeValue	The HRangeValue class provides a slider, or range control.
HScene	An HScene is a container representing the displayable area on-screen within which the application can display itself and thus interact with the user.
HSceneFactory	The HSceneFactory class provides a generic mechanism for an application to request HScene resources from a (conceptual) window management system.
HSceneTemplate	The HSceneTemplate class is used to obtain an HScene subject to a variety of constraints.
HScreen	This class describes the final output composition of a device.
HScreenConfigTemplate	This class describes a configuration of a screen device in terms of various properties and their importance to the application.
HScreenConfiguration	The HScreenConfiguration class describes the characteristics (settings) of an HScreenDevice .
HScreenDevice	An instance of the HScreen class represents a single independent video output signal from a device.
HScreenPoint	HScreenPoint denotes a screen location expressed as a relative value of the screen dimensions.
HScreenRectangle	HScreenRectangle denotes a screen area expressed as a relative value of the screen dimensions.
HSinglelineEntry	The HSinglelineEntry class is used to receive a single line of alphanumeric entry from the user and can also be used for password input.
HSinglelineEntryLook	The HSinglelineEntryLook class is used by the HSinglelineEntry component to display the entering of text.
HSound	The HSound class is used to represent an audio clip.
HStaticAnimation	The HStaticAnimation class is a user interface component used to display a sequence of images as an animation, but does *not* permit the user to navigate (focus) upon it.
HStaticIcon	This class creates an HStaticIcon (a graphical image).
HStaticRange	The HStaticRange component is used for displaying a value which is within a fixed range.

Class Summary

HStaticText	The HStaticText class is used to define a piece of textual content (label) that cannot be navigated to.
HStillImageBackgroundConfiguration	This class represents a background configuration which supports the installation of still images.
HText	The HText class is used to display static, read-only, navigable, text.
HTextButton	The HTextButton class is a conventional push- release textual button to be used for user actions.
HTextLook	The HTextLook class displays static read-only text on screen.
HToggleButton	The HToggleButton class creates a "check box", or with the support of the HToggleGroup class, "radio buttons".
HToggleGroup	HToggleButton within the same HToggleGroup will behave so that a maximum of one HToggleButton has switchable state true, as returned by getSwitchableState(), so as to achieve a "radio button" effect.
HVideoComponent	HVideoComponent is an opaque class encapsulating the presentation of a video source *within* an application, i.e.
HVideoConfigTemplate	The HVideoConfigTemplate class is used to obtain a valid HVideoConfiguration .
HVideoConfiguration	The HVideoConfiguration class describes the characteristics (settings) of an HVideoDevice .
HVideoDevice	The HVideoDevice class describes the logical video devices which can contribute to the appearance of a particular screen.
HVisible	The HVisible class is the base class for all non- interactive component, ie it only has a single (unfocused) state, denoted as NORMAL_STATE .

Exceptions

HConfigurationException	Thrown when an application requests an HScreenConfiguration that cannot be satisfied -- either because the HScreenConfiguration does not have the functionality, or because the requested HScreenConfiguration is otherwise invalid, e.g.
HInvalidLookException	A HInvalidLookException is an exception that is thrown when a a particular look is not compatable with the widget it has been associated with.
HMatteException	An HMatteException is an exception that is thrown when a Component is unable to support the desired HMatte effect.
HPermissionDeniedException	Thrown when an application calls a method which it does not have permission to do at that time.
HUIException	HUIException is a generic exception that indicates that the desired user-interface mechanism cannot be performed for some reason.

org.havi.ui
HActionable

Syntax
```
public interface HActionable extends HNavigable
```

All Known Subinterfaces:
HSwitchable

All Superinterfaces:
HNavigable

All Known Implementing Classes:
HTextButton, HGraphicButton

Description
This interface is implemented for all user interface components that are actionable (ie have three states: NORMAL_STATE , FOCUS_STATE and ACTION_STATE). Objects implementing the HActionable interface allow an ActionListener to be set on the component.

The HActionable reacts to focus events as an HNavigable .

The state transitions for an HActionable are as follows:

1. The HActionable is actioned when it receives an HUIEvent VK_ACTION or equivalent other Java AWT mechanism, e.g. mouse click.
2. The HActionable invokes its processEvent (processActionEvent)with an ACTION_PERFORMED java.awt.event.ActionEvent.
3. The HActionable modifies its interaction state as follows:

- NORMAL_STATE becomes NORMAL_ACTIONED_STATE .
- FOCUS_STATE and becomes ACTION_STATE .

1. The HActionable repaints itself to show any visual change in appearance, due to the associated HLook .
2. Any HSound associated with the HActionable (via setActionSound(HSound)) is played. Note that it is not guaranteed that the HSound will be played to completion.
3. Any ActionListeners are then called. The HActionable should block until all ActionListeners have returned, so that the HActionable reflects the status of the ActionListeners.
4. The HActionable modifies its interaction state as follows:

- NORMAL_ACTIONED_STATE becomes NORMAL_STATE .
- ACTION_STATE becomes FOCUS_STATE and.

1. The HActionable repaints itself to show any visual change in appearance, due to the associated HLook .

This is the only mechanism by which a component implementing HActionable may achieve the states ACTION_STATE or NORMAL_ACTIONED_STATE .

Subclasses of java.awt.Component which implement HActionable should by default enable both java.awt.event.FocusEvents and java.awt.event.ActionEvents. Whilst subclasses of java.awt.Component implementing HNavigable may enable additional java.awt.AWTEvents, applications should assume that such classes only generate java.awt.event.FocusEvents and java.awt.event.ActionEvent, and should use the standard AWT mechanisms to enable additional events to be generated, if required.

In particular, the following classes implementing HNavigable should all generate both java.awt.event.FocusEvent's and java.awt.event.HActionEvent's.

- HGraphicButton
- HTextButton
- HToggleButton
- HListElement

Classes implementing HActionable should also implement an additional protected method, as follows:

```
protected void processActionEvent(java.awt.event.ActionEvent evt)
```

in order to process java.awt.event.ActionEvents.

Methods

addActionListener(ActionListener)

```
public void addActionListener(java.awt.event.ActionListener l)
```

Adds the specified java.awt.ActionListener to receive java.awt.event.ActionEvent's from this object.

Parameters:

`l` - the ActionListener.

getActionCommand()

```
public java.lang.String getActionCommand()
```

Gets the command name for the java.awt.event.ActionEvent fired by this object.

Returns:

A String representing the command name of the action event fired by this object.

See Also:

`java.awt.event.ActionEvent.getActionCommand()`

getActionSound()

```
public HSound getActionSound()
```

Get the sound associated with the action event.

Returns:

The sound played when the component is actioned.

removeActionListener(ActionListener)

```
public void removeActionListener(java.awt.event.ActionListener l)
```

Removes the specified java.awt.ActionListener so that it no longer receives java.awt.event.Action-Event's from this object. If the specified listener is not registered, the method has no effect.

Parameters:

`l` - the ActionListener.

setActionCommand(String)

```
public void setActionCommand(java.lang.String command)
```

Sets the command name for the java.awt.event.ActionEvent fired by this object.

Parameters:

`command` - a string used to set the objects action command.

See Also:

`java.awt.event.ActionEvent.getActionCommand()`

setActionSound(HSound)

```
public void setActionSound(HSound sound)
```

Associate a sound with actioning this component.

Parameters:

`sound` - the sound to be played, when the component is actioned. If sound content is already set, the original content is replaced. To remove the sound specify a null HSound .

org.havi.ui
HActionInputPreferred

Syntax
```
public interface HActionInputPreferred
```

All Known Implementing Classes:
HGraphicButton, HListElement, HTextButton

Description
A component which implements HActionInputPreferred indicates that it is intended to be actioned.

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

org.havi.ui
HAdjustmentInputPreferred

Syntax
```
public interface HAdjustmentInputPreferred
```

All Known Implementing Classes:
HRangeValue, HListGroup

Description
A java.awt.Component which implements HAdjustmentInputPreferred indicates that this component does not expect to receive all subclasses of java.awt.event.KeyEvent, but simply expects the HUIEvent adjustment input events VK_ADJUST_MORE and VK_ADJUST_LESS . The system must provide a means of generating both VK_ADJUST_MORE and VK_ADJUST_LESS .

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

org.havi.ui
HAnimateEffect

Syntax
```
public interface HAnimateEffect
```

All Known Implementing Classes:
HStaticAnimation, HFlatEffectMatte, HImageEffectMatte

Description
The HAnimateEffect interface defines effect constants and controls for time-varying animations.

Implementations of HAnimateEffect should have the following default behaviours:

- By default the HAnimateEffect should be stopped. Hence, to start an HAnimateEffect the start() method must be explicitly invoked. This mechanism allows for animations that are programatically controlled, eg via the setPosition(int) method.
- By default the position for rendering should be the first image in the sequence, ie 0.
- By default the play mode should be PLAY_REPEATING .
- By default the repeat count should be REPEAT_INFINITE .
- The default rendering should simply display the single image at the current position of the animation within the sequence.

Fields

PLAY_ALTERNATING
```
public static final int PLAY_ALTERNATING
```
Indicates that the animation should be played in a repeating loop, alternating between the forward and reverse direction.

The images are rendered in the same order that they are present in the sequence (array)

0, 1, 2, 3, ... length-2, length-1

If the animation has not repeated sufficiently, then the rendering of the sequence is reversed. I.e. the images are rendered in the order

length-2, length-3, ... 1, 0

If the animation has not repeated sufficiently, then the rendering of the sequence is reversed (again) back to a forwards direction. I.e. the images are rendered in the order

1, 2, 3, ... length-2, length-1

Each rendering of the sequence of images forwards or backwards, should be considered as a single "repeat".

Note that when the sequence repeats, the last image (first image) is not rendered consecutively, ie twice.

PLAY_REPEATING
```
public static final int PLAY_REPEATING
```
Indicates that the animation should be played forwards (in a repeating loop).

The images are rendered in the same order that they are present in the sequence (array).

0, 1, 2, 3, ... length-1

If the animation has not repeated sufficiently, then the rendering of the sequence is restarted from the first image. I.e. the images will continue to be rendered in the order

0, 1, 2, 3, ... length-1

Each rendering of the sequence of images 0 to (length-1), should be considered as a single "repeat".

REPEAT_INFINITE

```
public static final int REPEAT_INFINITE
```

This value, when passed to setRepeatCount, indicates that the animation shall repeat until the stop() method is invoked.

Methods

getDelay()

```
public int getDelay()
```

Gets the presentation delay for the HAnimateEffect .

Returns:
the presentation delay in increments of 0.1 seconds.

getPlayMode()

```
public int getPlayMode()
```

Gets the playing mode for an HAnimateEffect .

Returns:
the playing mode for an HAnimateEffect .

getPosition()

```
public int getPosition()
```

Get the position at which the HAnimateEffect is displaying content.

Returns:
the index to the content to be displayed, 0 <= position < length.

getRepeatCount()

```
public int getRepeatCount()
```

Gets the number of times that an HAnimateEffect is to be played.

Returns:
the number of times that an HAnimateEffect is to be played. The returned value shall be greater than zero, or REPEAT_INFINITE .

isAnimated()

```
public boolean isAnimated()
```

This method indicates the animation (running) state of the HAnimateEffect .

Returns:

true if the HAnimateEffect is running, i.e. the start method has been invoked - false otherwise.

setDelay(int)

```
public void setDelay(int count)
```

Sets the delay between the presentation of successive content.

After calling setDelay(int) on a currently playing HAnimateEffect , there is no guarantee that one or more frames will not be displayed using the previous delay until the new delay value takes effect.

Parameters:

`count` - the content presentation delay in increments of 0.1 seconds. If count is less than one "unit", then it shall be treated as if it were a delay of one "unit", i.e. 0.1 seconds.

setPlayMode(int)

```
public void setPlayMode(int mode)
```

Sets the playing mode for an HAnimateEffect .

Parameters:

`mode` - the playing mode for an HAnimateEffect .

setPosition(int)

```
public void setPosition(int position)
```

Set the HAnimateEffect to display the content at the specified position

Parameters:

`position` - the member of the content array to be used in compositing the index to the content to be displayed. If position is < 0, then the 0 array element is displayed, if position >= length, then the [length-1] array element will be used.

setRepeatCount(int)

```
public void setRepeatCount(int count)
```

Sets the number of times that an HAnimateEffect should be played.

Parameters:

`count` - the number of times that an HAnimateEffect should be played. Valid values of the repeat count are one or more, and REPEAT_INFINITE . Invalid values should be treated as a repeat count of one.

start()

```
public void start()
```

This method starts the HAnimateEffect playing.

stop()

```
public void stop()
```

This method indicates that the running HAnimateEffect should be stopped. After calling this method, there is no guarantee that one or more frames will not be displayed before the animation actually stops playing.

org.havi.ui
HAnimateLook

Syntax

```
public class HAnimateLook implements HLook

java.lang.Object
  |
  +--org.havi.ui.HAnimateLook
```

All Implemented Interfaces:

java.lang.Cloneable, HLook

Description

This class defines the HAnimateLook of an HVisible and is the default HLook that is used by HStaticAnimation and its subclasses. The HAnimateLook will be provided by the platform and the exact way in which it is rendered will be platform dependant.

The HAnimateLook uses the content set on an HVisible , using the setAnimateContent(Image[], int) method. Its content related behaviour is as follows:

- If the sequence is empty then the animation shall be treated as a completely transparent area of specific width and height within its enclosing Container --- simply for the purposes of layout management (if applicable).
- If there is only one image referenced in the sequence then that image is rendered statically in a similar manner to HGraphicLook .
- If there is more than one image referenced in the sequence then these are rendered in sequence, giving the effect of an animation.

If a referenced image is inaccessible, then it shall be skipped. If no images are accessible, then the animation shall be treated as a completely transparent area of specific width and height within its enclosing Container --- for the purposes of layout management (if applicable).

The HAnimateLook is not required to present consecutive images in the animation with the delay specified in its HStaticAnimation . For example, if the time taken to retrieve or render an image is longer than the delay, then then it shall be rendered as soon as possible.

Default parameter values exposed in the constructors

None.

Default values not exposed in the constructors

Description	Default value	Set method	Get method
Scaling mode	NO_SCALING	setResizeMode(int)	getResizeMode()

See Also:

HStaticAnimation

Fields

NO_SCALING

```
public static final int NO_SCALING
```

This value indicates that the look should not attempt to resize the content to fit the widget.

SCALE_ARBITRARY

```
public static final int SCALE_ARBITRARY
```

This value indicates that the look should resize the content to fit the widget. Aspect ratios need not be preserved.

SCALE_PRESERVE

```
public static final int SCALE_PRESERVE
```

This value indicates that the look should resize the content to fit the widget while preserving the aspect ratio of the content. Areas of the widget that are not filled by the content will be look dependent.

Constructors

HAnimateLook()

```
public HAnimateLook()
```

Creates an Animate Look.

Methods

getHorizontalBorderSpacing()

```
public int getHorizontalBorderSpacing()
```

Specified By:
getHorizontalBorderSpacing() in interface HLook

See Also:
HLook

getMaximumSize(HVisible)

```
public java.awt.Dimension getMaximumSize(HVisible visible)
```

Specified By:
getMaximumSize(HVisible) in interface HLook

See Also:
HLook

getMinimumSize(HVisible)

```
public java.awt.Dimension getMinimumSize(HVisible visible)
```

Specified By:
getMinimumSize(HVisible) in interface HLook

See Also:
HLook

getPreferredSize(HVisible)

```
public java.awt.Dimension getPreferredSize(HVisible visible)
```

Specified By:
getPreferredSize(HVisible) in interface HLook

See Also:
HLook

getResizeMode()

```
public int getResizeMode()
```

Gets the content resize mode.

Returns:
the current resize mode

getVerticalBorderSpacing()

```
public int getVerticalBorderSpacing()
```

Specified By:
getVerticalBorderSpacing() in interface HLook

See Also:
HLook

setHorizontalBorderSpacing(int)

```
public void setHorizontalBorderSpacing(int width)
```

Specified By:
setHorizontalBorderSpacing(int) in interface HLook

See Also:
HLook

setResizeMode(int)

```
public boolean setResizeMode(int mode)
```

Sets the content resize mode.

Parameters:
`mode` - the desired mode (one of NO_SCALING, SCALE_PRESERVE or SCALE_ARBITRARY)

Returns:
false if the mode is not supported, true otherwise.

setVerticalBorderSpacing(int)

```
public void setVerticalBorderSpacing(int width)
```

Specified By:
setVerticalBorderSpacing(int) in interface HLook

See Also:
HLook

showLook(Graphics, HVisible, int)

```
public void showLook(java.awt.Graphics g, HVisible visible, int state)
```

Specified By:
showLook(Graphics, HVisible, int) in interface HLook

See Also:
HLook

org.havi.ui
HAnimation

Syntax

```
public class HAnimation extends HStaticAnimation implements HNavigable
```

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--HComponent
              |
              +--HVisible
                    |
                    +--HStaticAnimation
                          |
                          +--org.havi.ui.HAnimation
```

All Implemented Interfaces:

HAnimateEffect, HMatteLayer, HNavigable, HNoInputPreferred, HState, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

The HAnimation class is a user interface component used to display a sequence of images (as HStaticAnimation) which additionally enables a user to navigate (focus) upon it. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	Width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
height	Height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Parameter	Description	Default value	Set method	Get method
images	The array of images to be used as the content for both the NORMAL_STATE and FOCUS_STATE states of this component.	null	setAnimateContent(Image[], int)	getAnimateContent(int)
imagesNormal	The array of images to be used as the content for the NORMAL_STATE state of this component.	null	setAnimateContent(Image[], int)	getAnimateContent(int)
imagesFocused	The array of images to be used as the content for the FOCUS_STATE state of this component.	null	setAnimateContent(Image[], int)	getAnimateContent(int)
delay	The delay between the presentation of successive content in the animation.	1 (i.e. 0.1 seconds)	setDelay(int)	getDelay()
repeatCount	The number of times that the animation is to be played.	REPEAT_INFINITE	setRepeatCount(int)	getRepeatCount()
playMode	The playing mode for the animation.	PLAY_REPEATING	setPlayMode(int)	getPlayMode()

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The default "look" for this class.	A platform specific HAnimateLook	setDefaultLook(HAnimateLook)	getDefaultLook()
The "look" for this object.	The HAnimateLook returned from HAnimation.getDefaultLook when this object was created.	setLook(HLook)	getLook()
The gain focus sound.	null	setGainFocusSound(HSound)	getGainFocusSound()
The lose focus sound.	null	setLoseFocusSound(HSound)	getLoseFocusSound()

Description	Default value	Set method	Get method
The intial piece of content to be presented, i.e. its position in the content array.	0	setPosition(int)	getPosition()
By default the animation should be stopped. Hence, to start the animation its start method must be explicitly invoked. This mechanism allows for animations that are programatically controlled, eg via the setPosition method.	"stopped"	start() / stop()	isAnimated()

See Also:

HStaticAnimation, HNavigable

Constructors

HAnimation()

```
public HAnimation()
```

Creates an HAnimation object. See the class description for details of constructor parameters and default values.

HAnimation(Image[], Image[], int, int, int)

```
public HAnimation(java.awt.Image[] imagesNormal, java.awt.Image[] imagesFocused, int delay,
            int playMode, int repeatCount)
```

Creates an HAnimation object. See the class description for details of constructor parameters and default values.

HAnimation(Image[], Image[], int, int, int, int, int, int, int)

```
public HAnimation(java.awt.Image[] imagesNormal, java.awt.Image[] imagesFocused, int delay,
            int playMode, int repeatCount, int x, int y, int width, int height)
```

Creates an HAnimation object. See the class description for details of constructor parameters and default values.

HAnimation(Image[], int, int, int)

```
public HAnimation(java.awt.Image[] images, int delay, int playMode, int repeatCount)
```

Creates an HAnimation object. See the class description for details of constructor parameters and default values.

HAnimation(Image[], int, int, int, int, int, int, int)

```
public HAnimation(java.awt.Image[] images, int delay, int playMode, int repeatCount, int x,
            int y, int width, int height)
```

Creates an HAnimation object. See the class description for details of constructor parameters and default values.

Methods

getDefaultLook()

```
public static HAnimateLook getDefaultLook()
```

Returns the currently set default look for HAnimation components.

Returns:
The HLook that is used by default when creating a new HAnimation component.

getGainFocusSound()

```
public HSound getGainFocusSound()
```

Specified By:
getGainFocusSound() in interface HNavigable

See Also:
HNavigable

getLoseFocusSound()

```
public HSound getLoseFocusSound()
```

Specified By:
getLoseFocusSound() in interface HNavigable

See Also:
HNavigable

getMove(int)

```
public HNavigable getMove(int keyCode)
```

Specified By:
getMove(int) in interface HNavigable

See Also:
HNavigable

isFocusTraversable()

```
public boolean isFocusTraversable()
```

By default classes implementing HNavigable are focus-traversable.

Overrides:
isFocusTraversable() in class HVisible

Returns:
true

See Also:
`java.awt.Component.isFocusTraversable()`

isSelected()

```
public boolean isSelected()
```

Specified By:
isSelected() in interface HNavigable

See Also:
HNavigable

processEvent(AWTEvent)

```
protected void processEvent(java.awt.AWTEvent evt)
```

It is a valid implementation option for HAnimation to override the protected processEvent method.

Overrides:
java.awt.Component.processEvent(java.awt.AWTEvent) in class java.awt.Component

See Also:
`java.awt.Component.processEvent(AWTEvent)`

processFocusEvent(FocusEvent)

```
protected void processFocusEvent(java.awt.event.FocusEvent evt)
```

It is a valid implementation option for HAnimation to override the protected processFocusEvent method.

Overrides:
java.awt.Component.processFocusEvent(java.awt.event.FocusEvent) in class java.awt.Component

See Also:
`java.awt.Component.processFocusEvent(FocusEvent)`

requestFocus()

```
public void requestFocus()
```

Specified By:
requestFocus() in interface HNavigable

Overrides:
java.awt.Component.requestFocus() in class java.awt.Component

See Also:
`java.awt.Component.requestFocus()`

setDefaultLook(HAnimateLook)

```
public static void setDefaultLook(HAnimateLook hlook)
```

Sets the default look for all HAnimation Components.

Parameters:
`hlook` - The HLook that will be used by default when creating a new HAnimation component.

Throws:
HInvalidLookException - If the HLook is not an HAnimateLook .

setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
            HNavigable right)
```

Specified By:
setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) in interface HNavigable

See Also:
HNavigable

setGainFocusSound(HSound)

```
public void setGainFocusSound(HSound sound)
```

Specified By:
setGainFocusSound(HSound) in interface HNavigable

See Also:
HNavigable

setLoseFocusSound(HSound)

```
public void setLoseFocusSound(HSound sound)
```

Specified By:
setLoseFocusSound(HSound) in interface HNavigable

See Also:
HNavigable

setMove(int, HNavigable)

```
public void setMove(int keyCode, HNavigable target)
```

Specified By:
setMove(int, HNavigable) in interface HNavigable

See Also:
HNavigable

org.havi.ui
HBackgroundConfigTemplate

Syntax

```
public class HBackgroundConfigTemplate extends HScreenConfigTemplate

java.lang.Object
  |
  +--HScreenConfigTemplate
        |
        +--org.havi.ui.HBackgroundConfigTemplate
```

Description

The HBackgroundConfigTemplate class is used to obtain a valid HBackgroundConfiguration . An application instantiates one of these objects and then sets all non-default attributes as desired. The getBestConfiguration(HBackgroundConfigTemplate) method found in the HBackgroundDevice class is then called with this HBackgroundConfigTemplate . A valid HBackgroundConfiguration is returned that meets or exceeds what was requested in the HBackgroundConfigTemplate .

This class this may be subclassed to support define additional properties of backgrounds which may be requested by applications. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default values exposed in the constructors

None.

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Fields

CHANGEABLE_SINGLE_COLOR

```
public static final int CHANGEABLE_SINGLE_COLOR
```

A value for use in the preference field of the setPreference(int, int) method in the HBackgroundConfigTemplate that indicates that a single colour background is requested where that single colour can be changed by applications.

STILL_IMAGE

```
public static final int STILL_IMAGE
```

A value for use in the preference field of the setPreference(int, int) method in the HBackgroundConfigTemplate that indicates that a background which can support still images is requested. Where backgrounds supporting this feature are returned, they are returned as objects of the HStillImageBackgroundConfiguration class.

Constructors

HBackgroundConfigTemplate()

```
public HBackgroundConfigTemplate()
```

Creates an HBackgroundConfigTemplate object.

org.havi.ui
HBackgroundConfiguration

Syntax
```
public class HBackgroundConfiguration extends HScreenConfiguration

java.lang.Object
  |
  +--HScreenConfiguration
        |
        +--org.havi.ui.HBackgroundConfiguration
```

Direct Known Subclasses:
HStillImageBackgroundConfiguration

Description
The HBackgroundConfiguration class describes the characteristics (settings) of an HBackgroundDevice . There can be many HBackgroundConfiguration objects associated with a single HBackgroundDevice .

The basic background configuration supports backgrounds of a single colour. More sophisticated backgrounds can be supported by defining new classes inheriting from this class. Where a device has a single non- changeable background colour, this class will provide applications the ability to read that colour however all attempts to reserve control of the background will fail.

Default values exposed in the constructors

None.

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

See Also:
HBackgroundDevice

Constructors

HBackgroundConfiguration()
```
protected HBackgroundConfiguration()
```
It is not intended that applications should directly construct HBackgroundConfiguration objects. Creates an HBackgroundConfiguration object. See the class description for details of constructor parameters and default values.

Methods

getColor()

```
public java.awt.Color getColor()
```

Obtain the current colour of this background. This method may be called without ownership of the resource. The value returned is not guaranteed to be the value set in the last call to setColor(Color) since platforms may offer a reduced colour space for backgrounds and the actual value used will be returned.

Returns:
the current Color

getConfigTemplate()

```
public HBackgroundConfigTemplate getConfigTemplate()
```

Returns an HBackgroundConfigTemplate object that describes and uniquely identifies this HBackgroundConfiguration . Hence, the following sequence should return the original HBackgroundConfiguration .

```
HBackgroundDevice.getBestMatch(HBackgroundConfiguration.getConfigTemplate())
```

Properties that are implemented in the HBackgroundConfiguration will return REQUIRED priority, features that are not implemented in the HBackgroundConfiguration will return REQUIRED_NOT priority.

Returns:
an HBackgroundConfigTemplate object which both describes and uniquely identifies this HBackgroundConfiguration .

getDevice()

```
public HBackgroundDevice getDevice()
```

Returns the HBackgroundDevice associated with this HBackgroundConfiguration .

Returns:
the HBackgroundDevice object that is associated with this HBackgroundConfiguration .

setColor(Color)

```
public void setColor(java.awt.Color color)
```

Set the current colour of this background. On platforms where there is a sub-class of java.awt.Color supporting transparency of any kind, passing an object representing a non-opaque colour is illegal. Platforms with a limited colour resolution for backgrounds may approximate this value to the nearest available. The getColor() method will return the actual value used.

Parameters:
`color` - the colour to be used for the background

Throws:
HPermissionDeniedException - if this HBackgroundDevice does not have the right to control the background

`{@link` - org.havi.ui.HConfigurationException HConfigurationException} if the colour specified is illegal for this platform.

HConfigurationException

org.havi.ui
HBackgroundDevice

Syntax

```
public class HBackgroundDevice extends HScreenDevice

java.lang.Object
  |
  +--HScreenDevice
        |
        +--org.havi.ui.HBackgroundDevice
```

All Implemented Interfaces:

org.davic.resources.ResourceProxy, org.davic.resources.ResourceServer

Description

This class represents the ultimate background of a screen. The background is the very back of the video / graphics composition stack. It can potentially cover the entire area of a screen. Where a device supports multiple applications on screen at the same time (or even a window manager), the background is not constrained by any particular application or window. The right to control the background of a screen is a scarce resource and managed as such.

Default values exposed in the constructors

None.

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Constructors

HBackgroundDevice()

```
protected HBackgroundDevice()
```

It is not intended that applications should directly construct HBackgroundDevice objects. Creates an HBackgroundDevice object. See the class description for details of constructor parameters and default values.

Methods

getBestConfiguration(HBackgroundConfigTemplate)

```
public HBackgroundConfiguration getBestConfiguration(HBackgroundConfigTemplate hbc)
```

Returns the "best" configuration possible that passes the criteria defined in this HBackgroundConfigTemplate .

Best in this sense means satisfying the preferences in the config template as follows based on the priority (as supplied to HScreenConfigTemplate.setPreference())

1. satisfying all the preferences in that config template whose was REQUIRED
2. excluding configurations with priorites which were REQUIRED_NOT
3. satisfying as many as possible of the preferences whose priority was PREFERRED .
4. Satisfying as few as possible of the preferences whose priority was PREFERRED_NOT .

Parameters:
`hbc` - - an HBackgroundConfigTemplate object used to obtain a valid HBackgroundConfiguration

Returns:
an HBackgroundConfiguration object that passes the criteria defined in the specified HGraphicsConfigTemplate

getBestConfiguration(HBackgroundConfigTemplate[])

```
public HBackgroundConfiguration getBestConfiguration(HBackgroundConfigTemplate[] hbcta)
```

Returns the "best" configuration possible that passes the criteria defined in one of the HBackgroundConfigTemplate objects within the specified array. The HBackgroundTemplate objects should be considered for matching in priority order from 0 to (hbcta.length - 1).

Best in this sense means satisfying the preferences in the config template as follows based on the priority (as supplied to HScreenConfigTemplate.setPreference())

1. satisfying all the preferences in that config template whose was REQUIRED
2. excluding configurations with priorites which were REQUIRED_NOT
3. satisfying as many as possible of the preferences whose priority was PREFERRED .
4. Satisfying as few as possible of the preferences whose priority was PREFERRED_NOT .

Parameters:
`hbcta` - the HBackgroundConfigTemplate array used to obtain a valid HBackgroundConfiguration .

Returns:
an HBackgroundConfiguration that passes the criteria defined in one of the HBackgroundConfigTemplate objects within the specified array

getConfigurations()

```
public HBackgroundConfiguration[] getConfigurations()
```

Returns all of the HBackgroundConfiguration objects associated with this HBackgroundDevice .

Returns:
an array of HBackgroundConfiguration objects

See Also:
HBackgroundConfiguration

getCurrentConfiguration()

```
public HBackgroundConfiguration getCurrentConfiguration()
```

Returns the current HBackgroundConfiguration for this HBackgroundDevice .

Returns:
the current HBackgroundConfiguration for this HBackgroundDevice .

See Also:
HBackgroundConfiguration

getDefaultConfiguration()

```
public HBackgroundConfiguration getDefaultConfiguration()
```

Returns the default HBackgroundConfiguration associated with this HBackgroundDevice . This (single) default configuration should correspond to some well-behaved settings for the device, such as, a minimal configuration or factory preset settings.

Returns:
the default HBackgroundConfiguration of this HBackgroundDevice

See Also:
HBackgroundConfiguration

setBackgroundConfiguration(HBackgroundConfiguration)

```
public boolean setBackgroundConfiguration(HBackgroundConfiguration hbc)
```

Set the background configuration for the device.

Parameters:
`hbc` - the HBackgroundConfiguration to which this device should be set.

Returns:
A boolean indicating whether the configuration could be applied successfully. If the configuration could not be applied successfully, the configuration after this method may not match the configuration of the device prior to this method being called --- applications should take steps to determine whether a partial change of settings has been made.

Throws:
`SecurityException` - if the application does not have sufficient rights to set the configuration for this device.

HPermissionDeniedException - (HPermissionDeniedException) if the application does not currently have the right to set the configuration for this device.

HConfigurationException - (HConfigurationException) if the specified configuration is not valid for this device.

org.havi.ui
HBackgroundImage

Syntax
```
public class HBackgroundImage

java.lang.Object
  |
  +--org.havi.ui.HBackgroundImage
```

Description
This class represents a background image. Images of this class can be used as full screen backgrounds outside the java.awt framework. Default values exposed in the constructors: None.

Default values not exposed in the constructors

None.

Constructors

HBackgroundImage(String)

```
public HBackgroundImage(java.lang.String filename)
```

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

Parameters:
`filename` - the name of the file to use as the source of data

Methods

flush()

```
public void flush()
```

Flush all the resources used by this image. This includes any pixel data being cached as well as all underlying system resources used to store data or pixels for the image. After calling this method the image is in a state similar to when it was first created without any load method having been called. When this method is called, the image shall not be in use by an application. Resources related to any HBackgroundDevice are not released.

getHeight()

```
public int getHeight()
```

Determines the height of the image. This is returned in pixels as defined by the format of the image concerned. If this information is not known when this method is called then -1 is returned.

Returns:
the width of the image

getWidth()

```
public int getWidth()
```

Determines the width of the image. This is returned in pixels as defined by the format of the image concerned. If this information is not known when this method is called then -1 is returned.

Returns:
the width of the image

load(HBackgroundImageListener)

```
public void load(HBackgroundImageListener l)
```

Load the data for this object. This method is asynchronous. The completion of data loading is reported through the listener provided.

Parameters:
`l` - the listener to call when loading of data is completed.

org.havi.ui
HComponent

Syntax

```
public abstract class HComponent extends java.awt.Component implements HMatteLayer

java.lang.Object
  |
  +--java.awt.Component
        |
        +--org.havi.ui.HComponent
```

Direct Known Subclasses:

HVideoComponent, HVisible

All Implemented Interfaces:

HMatteLayer, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

The HComponent class extends the java.awt.Component class by implementing the HMatteLayer interface.

Default values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
height	height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Default values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte (HMatte).	none	setMatte(HMatte)	getMatte()

Constructors

HComponent()

```
public HComponent()
```

Creates an HComponent object. See the class description for details of constructor parameters and default values.

HComponent(int, int, int, int)

```
public HComponent(int x, int y, int width, int height)
```

Creates an HComponent object. See the class description for details of constructor parameters and default values.

Methods

getMatte()

```
public HMatte getMatte()
```

Specified By:
getMatte() in interface HMatteLayer

See Also:
HMatteLayer

isDoubleBuffered()

```
public boolean isDoubleBuffered()
```

Returns true if all the drawing done during the update and paint methods for this specific HComponent object is automatically double buffered.

Overrides:
java.awt.Component.isDoubleBuffered() in class java.awt.Component

Returns:
true if all the drawing done during the update and paint methods for this specific HComponent object is automatically double buffered. The default value for the double buffering setting is platform-specific.

isOpaque()

```
public boolean isOpaque()
```

Returns true if the entire HComponent area, as given by the java.awt.Component#getBounds method, is fully opaque, i.e. its paint method (or surrogate methods) guarantee that all pixels are painted in an opaque Color.

Returns:

true if all the pixels with the java.awt.Component#getBounds method are fully opaque, i.e. its paint method (or surrogate methods) guarantee that all pixels are painted in an opaque Color, otherwise false.

By default, the return value is false. The return value should be overridden by subclasses that can guarantee full opacity. The consequences of an invalid overridden value are implementation specific.

setMatte(HMatte)

```
public void setMatte(HMatte m)
```

Specified By:

setMatte(HMatte) in interface HMatteLayer

See Also:

HMatteLayer

org.havi.ui
HConfigurationException

Syntax
```
public class HConfigurationException extends java.lang.Exception

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.havi.ui.HConfigurationException
```

All Implemented Interfaces:
java.io.Serializable

Description
Thrown when an application requests an HScreenConfiguration that cannot be satisfied -- either because the HScreenConfiguration does not have the functionality, or because the requested HScreenConfiguration is otherwise invalid, e.g. it is a HScreenConfiguration due to a different HScreenDevice than the one it is being applied to. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter value parameter values exposed in the constructors: None.

Default values not exposed in the constructors

None.

Constructors

HConfigurationException()
```
public HConfigurationException()
```
Default constructor for the exception

HConfigurationException(String)
```
public HConfigurationException(java.lang.String reason)
```
Constructor for the exception with a specified reason

Parameters:
`reason` - the reason why the exception was raised

org.havi.ui
HContainer

Syntax

```
public class HContainer extends java.awt.Container implements HMatteLayer

java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--org.havi.ui.HContainer
```

Direct Known Subclasses:

HDialog, HListGroup, HScene

All Implemented Interfaces:

HMatteLayer, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

The HContainer class extends the java.awt.Container class by implementing the HMatteLayer interface and providing additional z-ordering capabilities, which are required since components in the HAVi user-interface are explicitly allowed to overlap each other. Note that these z-ordering capabilities (addBefore, addAfter, pop, popInFrontOf, popToFront, push, pushBehind and pushToBack) must be implemented by (implicitly) reordering the child Components within the HContainer , so that the standard AWT convention that Z-ordering is defined as the order in which Components are added to a given Container is maintained. For example, one implementation of popToFront might be to make the specified Component become the first Component added to the parent Container by removing all Components from that Container, adding the specified Container first, and then adding the remaining Components in their current relative order to that Container.

Default values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	Width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Parameter	Description	Default value	Set method	Get method
height	Height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Default values not exposed in the constructors

Description	Default value	Set method	Get method
Associated matte (HMatte).	none	setMatte(HMatte)	getMatte()
LayoutManager	null (in contrast to java.awt.Container)	java.awt.Container#setLayout	java.awt.Container#getLayout

Constructors

HContainer()

```
public HContainer()
```

Creates an HContainer object. See the class description for details of constructor parameters and default values.

HContainer(int, int, int, int)

```
public HContainer(int x, int y, int width, int height)
```

Creates an HContainer object. See the class description for details of constructor parameters and default values.

Methods

addAfter(Component, Component)

```
public synchronized java.awt.Component addAfter(java.awt.Component component, java.awt.
            Component front)
```

Adds a java.awt.Component to this HContainer directly behind a previously added java.awt.Component.

This method has effects in terms of the Z-ordering of java.awt.Components within a HContainer , and may implicitly also change their effective prior numeric ordering within that HContainer .

Parameters:

`component` - is the java.awt.Component to be added to the Scene

`front` - is the java.awt.Component, which component will be placed behind, i.e. front will be directly in front of the added java.awt.Component

Returns:

If the java.awt.Component is successfully added, then it will be returned from this call. If the java.awt.Component is not successfully added, e.g. front is not a java.awt.Component currently added to the HContainer , then null will be returned.

This method must be implemented in a thread safe manner.

addBefore(Component, Component)

```
public synchronized java.awt.Component addBefore(java.awt.Component component, java.awt.
            Component behind)
```

Adds a java.awt.Component to this HContainer directly in front of a previously added java.awt.Component.

This method has effects in terms of the Z-ordering of java.awt.Components within a HContainer , and may implicitly also change their effective prior numeric ordering within that HContainer .

Parameters:

`component` - is the java.awt.Component to be added to the Scene

`behind` - is the java.awt.Component, which component will be placed in front of, i.e. behind will be directly behind the added java.awt.Component

Returns:

If the java.awt.Component is successfully added, then it will be returned from this call. If the java.awt.Component is not successfully added, e.g. behind is not a java.awt.Component currently added to the HContainer , then null will be returned.

This method must be implemented in a thread safe manner

getMatte()

```
public HMatte getMatte()
```

Specified By:

getMatte() in interface HMatteLayer

See Also:

HMatteLayer

group()

```
public void group()
```

Groups the HContainer and its components.

See Also:

ungroup()

isDoubleBuffered()

```
public boolean isDoubleBuffered()
```

Returns true if all the drawing done during the update and paint methods for this specific HContainer object is automatically double buffered.

Overrides:

java.awt.Component.isDoubleBuffered() in class java.awt.Component

Returns:

true if all the drawing done during the update and paint methods for this specific HContainer object is automatically double buffered. The default value for the double buffering setting is platform-specific.

isGrouped()

```
public boolean isGrouped()
```

Tests whether the HContainer and its components are grouped.

Returns:
returns true if the HContainer and its components are grouped, false otherwise.

See Also:
group()

isOpaque()

```
public boolean isOpaque()
```

Returns true if the entire HContainer area, as given by the java.awt.Component#getBounds method, is fully opaque, i.e. its paint method (or surrogate methods) guarantee that all pixels are painted in an opaque Color.

Returns:
true if all the pixels with the java.awt.Component#getBounds method are fully opaque, i.e. its paint method (or surrogate methods) guarantee that all pixels are painted in an opaque Color, otherwise false.

By default, the return value is false. The return value should be overridden by subclasses that can guarantee full opacity. The consequences of an invalid overridden value are implementation specific.

pop(Component)

```
public boolean pop(java.awt.Component component)
```

Moves the specified java.awt.Component one component nearer in the Z-ordering, ie swapping it with the component that was directly in front of it.

Parameters:
`component` - The java.awt.Component to be moved.

Returns:
returns true on success, false on failure, for example if the java.awt.Component has yet to be added to the HContainer .

popInFrontOf(Component, Component)

```
public boolean popInFrontOf(java.awt.Component move, java.awt.Component behind)
```

Puts the specified java.awt.Component in front of another java.awt.Component in the HContainer z-order.

Parameters:
`move` - The java.awt.Component to be moved directly in front of the "behind" Component in the HContainer z-order.

`behind` - The java.awt.Component which the "move" Component should be placed directly in front of.

Returns:
returns true on success, false on failure, for example when either java.awt.Component has yet to be added to the HContainer .

popToFront(Component)

```
public boolean popToFront(java.awt.Component component)
```

Brings the specified java.awt.Component to the "front" of the HContainer z-order.

Parameters:
`component` - The java.awt.Component to bring to the "front" of the HContainer z-order.

Returns:
returns true on success, false on failure, for example when the java.awt.Component has yet to be added to the HContainer .

push(Component)

```
public boolean push(java.awt.Component component)
```

Moves the specified java.awt.Component one component further away in the Z-ordering, ie swapping it with the component that was directly behind it.

Parameters:
`component` - The java.awt.Component to be moved.

Returns:
returns true on success, false on failure, for example if the java.awt.Component has yet to be added to the HContainer .

pushBehind(Component, Component)

```
public boolean pushBehind(java.awt.Component move, java.awt.Component front)
```

Puts the specified java.awt.Component behind another java.awt.Component in the HContainer z-order.

Parameters:
`move` - The java.awt.Component to be moved directly behind the "front" Component in the HContainer z-order.

`front` - The java.awt.Component which the "move" Component should be placed directly in behind.

Returns:
returns true on success, false on failure, for example when either java.awt.Component has yet to be added to the HContainer .

pushToBack(Component)

```
public boolean pushToBack(java.awt.Component component)
```

Place the specified java.awt.Component at the "back" of the HContainer z-order.

Parameters:
`component` - The java.awt.Component to place at the "back" of the HContainer z-order.

Returns:
returns true on success, false on failure, for example when the java.awt.Component has yet to be added to the HContainer .

setMatte(HMatte)

```
public void setMatte(HMatte m)
```

Specified By:
setMatte(HMatte) in interface HMatteLayer

See Also:
HMatteLayer

ungroup()

```
public void ungroup()
```

Ungroups the HContainer and its components.

See Also:

group()

org.havi.ui
HDefaultTextLayoutManager

Syntax

```
public class HDefaultTextLayoutManager implements HTextLayoutManager

java.lang.Object
  |
  +--org.havi.ui.HDefaultTextLayoutManager
```

All Implemented Interfaces:

HTextLayoutManager

Description

The HDefaultTextLayoutManager provides the default text rendering mechanism for the HStaticText HText and HTextButton classes.

The HDefaultTextLayoutManager allows text to be aligned in both horizontal and vertical directions, or justified, etc. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
hAlign	The horizontal alignment of the text layout manager	CENTER_HORIZONTAL	setHAlign	getHAlign
vAlign	The vertical alignment of the text layout manager	CENTER_VERTICAL	setVAlign	getVAlign

Default values not exposed in the constructors

None.

Fields

BOTTOM_ALIGN

```
public static final int BOTTOM_ALIGN
```

The text string should be vertically aligned to the bottom of the specified area.

CENTER_HORIZONTAL

```
public static final int CENTER_HORIZONTAL
```

The text string should be centered horizontally.

CENTER_VERTICAL

```
public static final int CENTER_VERTICAL
```

The text string should be centered vertically.

HORIZONTAL_JUSTIFY

```
public static final int HORIZONTAL_JUSTIFY
```

The text string should be fully justified (horizontally).

LEFT_ALIGN

```
public static final int LEFT_ALIGN
```

The text string should be left aligned.

RIGHT_ALIGN

```
public static final int RIGHT_ALIGN
```

The text string should be right aligned

TOP_ALIGN

```
public static final int TOP_ALIGN
```

The text string should be vertically aligned to the top of the specified area.

VERTICAL_JUSTIFY

```
public static final int VERTICAL_JUSTIFY
```

The text string should be fully justified (vertically).

Constructors

HDefaultTextLayoutManager()

```
public HDefaultTextLayoutManager()
```

Creates an HDefaultTextLayoutManager object. See the class description for details of constructor parameters and default values.

HDefaultTextLayoutManager(int, int)

```
public HDefaultTextLayoutManager(int hAlign, int vAlign)
```

Creates an HDefaultTextLayoutManager object. See the class description for details of constructor parameters and default values.

Methods

getHAlign()

```
public int getHAlign()
```

Get the horizontal alignment.

Returns:
the horizontal alignment.

getVAlign()

```
public int getVAlign()
```

Get the vertical alignment.

Returns:
the vertical alignment.

render(String, Graphics, HVisible)

```
public void render(java.lang.String markedUpString, java.awt.Graphics g, HVisible v)
```

The HDefaultTextLayoutManager should use the java.awt.Graphics object and HVisible as a basis to determine the rendering.

The default behaviour if the specified font cannot be accessed is to replace it with the nearest builtin font. Each missing character is replaced with an "!" character.

Antialiasing behaviours are platform dependent.

Specified By:
render(String, Graphics, HVisible) in interface HTextLayoutManager

Parameters:
`markedUpString` - the marked-up string to parse, and render. The string of text to be laid out may be multi-line, where each line is separated by a "\\n" (0x0A).

`g` - the java.awt.Graphics object whose properties should be used as a basis for the rendering.

`v` - the HVisible object whose properties should be used as a basis for the rendering.

See Also:
HTextLayoutManager

setHAlign(int)

```
public void setHAlign(int hAlign)
```

Set the horizontal alignment.

Parameters:
`hAlign` - Horizontal alignment.

setVAlign(int)

```
public void setVAlign(int vAlign)
```

Set the vertical alignment.

Parameters:
`vAlign` - Vertical alignment.

org.havi.ui
HDialog

Syntax

```
public class HDialog extends HContainer

java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--HContainer
                    |
                    +--org.havi.ui.HDialog
```

All Implemented Interfaces:

HMatteLayer, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

The HDialog class extends the HContainer class to allow a modal dialog/pop-up to be displayed by the application. The HDialog is used by adding components to it, and then calling startDialog(Component, HScene, boolean) . A button will usually be added that has an associated ActionListener that calls endDialog(int) e.g. the OK and Cancel buttons.

If a background image is specified for the HDialog , it is assumed that this image will also provide the appearance of the frame bounding the HDialog , and therefore the drawDefaultFrame(Graphics) method will not be invoked, i.e. HDialog will not draw a rectangle around itself.

To allow maximum flexibility in implementation, no parent component to an HDialog should be accessible to applications. Interoperable applications should not use the getParent method in HDialog , since results are implementation dependent and valid implementations may generate a run-time error. In terms of delegation, the HDialog shall behave such that it has valid defaults defined for all characteristics, e.g. Font, foreground Color, background Color, ColorModel, Cursor and Locale, which should be equivalent to those for its corresponding HScene . The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Parameter	Description	Default value	Set method	Get method
width	Width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
height	Height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Constructors

HDialog()

```
public HDialog()
```

Creates an HDialog object. See the class description for details of constructor parameters and default values.

HDialog(int, int, int, int)

```
public HDialog(int x, int y, int width, int height)
```

Creates an HDialog object. See the class description for details of constructor parameters and default values.

Methods

drawDefaultFrame(Graphics)

```
protected void drawDefaultFrame(java.awt.Graphics g)
```

Called by the paint method to draw the default frame around the dialog. Override this method to change the appearance of the frame

Parameters:

`g` - the specified context to use for updating.

endDialog(int)

```
public void endDialog(int returnValue)
```

Closes the dialog. This function will return control back through startDialog with the value passed in as returnValue .

Subclasses of HDialog overriding endDialog must call super.endDialog to ensure that the HDialog subclass is started consistently.

Parameters:
`returnValue` - value returned to startDialog

getInsets()

```
public java.awt.Insets getInsets()
```

Returns current insets

Overrides:
java.awt.Container.getInsets() in class java.awt.Container

Returns:
the Insets of this HDialog

paint(Graphics)

```
public void paint(java.awt.Graphics g)
```

Called by the framework when the HDialog requires painting. The background colour of the dialog is first painted, and if a background image is specified it is also painted. All components of the HDialog will then be painted. If no background image is specified the method drawDefaultFrame(Graphics) will finally be called to draw a frame around the dialog.

Overrides:
java.awt.Container.paint(java.awt.Graphics) in class java.awt.Container

Parameters:
`g` - the specified context to use for updating.

setBackground(Image)

```
public void setBackground(java.awt.Image image)
```

Sets the background image for the HDialog . If a background image is provided, it is assumed to include any border decoration and therefore the drawDefaultFrame(Graphics) method will not be invoked, and hence no additional border decoration will be drawn.

setInsets(int, int, int, int)

```
public void setInsets(int top, int left, int bottom, int right)
```

Sets the insets (borders) of the HDialog

startDialog(Component, HScene, boolean)

```
public int startDialog(java.awt.Component initial, HScene scene, boolean urgent)
```

Invoking HDialog .startDialog will perform the following actions:

1. The HDialog is made visible.
2. The HDialog receives a FOCUS_GAINED java.awt.event.focusEvent
3. All user input events are directed to the HDialog , effectively disabling all widgets outside the dialog.
4. Short-Cut keys will still be processed by the HScene and therefore might need to be disabled by the application programmer before the startDialog(Component, HScene, boolean) method is called.
5. Sets focus to the initial widget in the dialog that has been specified to have focus, generating a FOCUS_GAINED java.awt.event.focusEvent on the widget.
6. The startDialog(Component, HScene, boolean) method does not return until an endDialog(int) is called.

7. When endDialog(int) is called the widget within the dialog that currently has focus receives a FOCUS_LOST java.awt.event.focusEvent
8. The dialog receives a FOCUS_LOST java.awt.event.focusEvent
9. The dialog is made invisible
10. The widget in the HScene that had focus before the startDialog(Component, HScene, boolean) method was invoked receives a FOCUS_GAINED java.awt.event.focusEvent
11. The startDialog(Component, HScene, boolean) method returns with a value that passed into the endDialog(int) method.

Subclasses of HDialog overriding startDialog(Component, HScene, boolean) must call super.startDialog to ensure that the HDialog subclass is started consistently.

Parameters:

`initial` - the component that will be given focus when the dialog is first displayed on screen.

`scene` - the HScene within which the HDialog should be presented.

`urgent` - a value of true indicates the application will request to be made visible and input focus from the navigation shell.

Returns:

the value passed into endDialog(int) .

org.havi.ui
HEmulatedGraphicsConfiguration

Syntax

```
public class HEmulatedGraphicsConfiguration extends HGraphicsConfiguration

java.lang.Object
  |
  +--HScreenConfiguration
        |
        +--HGraphicsConfiguration
              |
              +--org.havi.ui.HEmulatedGraphicsConfiguration
```

Description

A HEmulatedGraphicsConfiguration is a configuration for a "virtual" graphics device that may perform one or more emulations, e.g. in the ATSC context a HEmulatedGraphicsDevice might implement multiple HEmulatedGraphicsConfiguration , corresponding to each of the possible relationships to the high-definition display modes. The HEmulatedGraphicsConfiguration would be used to configure a device appropriately for rendering into, whilst mapping the emulated device onto the "true" physical display, e.g. by down-sampling to standard-definition display.

In essence the HEmulatedGraphicsConfiguration may be considered as a pair of HGraphicsConfiguration objects: one describing the configuration of the emulation and the second describing the corresponding configuration of the implementation.

Hence, an HGraphicsConfiguration may be considered as a special case of the HEmulatedGraphicsConfiguration class, where the emulation and implementation are equivalent. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

See Also:

HEmulatedGraphicsDevice, HGraphicsConfiguration

Constructors

HEmulatedGraphicsConfiguration()

```
protected HEmulatedGraphicsConfiguration()
```

It is not intended that applications should directly construct HEmulatedGraphicsConfiguration objects. Creates an HEmulatedGraphicsConfiguration object. See the class description for details of constructor parameters and default values.

Methods

getConfigTemplate()

```
public HGraphicsConfigTemplate getConfigTemplate()
```

Returns an HGraphicsConfigTemplate describing the virtual (emulation) characteristics of the HEmulatedGraphicsDevice .

Overridden method from HGraphicsConfiguration -- for an HEmulatedGraphicsConfiguration this returns a description of the emulation characteristics.

Overrides:
getConfigTemplate() in class HGraphicsConfiguration

Returns:
an HGraphicsConfigTemplate describing the virtual (emulation) characteristics of the HEmulatedGraphicsDevice .

See Also:
HGraphicsConfigTemplate, HGraphicsConfiguration, HEmulatedGraphicsDevice

getEmulation()

```
public HGraphicsConfigTemplate getEmulation()
```

Returns an HGraphicsConfigTemplate describing the virtual (emulation) characteristics of the HEmulatedGraphicsDevice .

Returns:
an HGraphicsConfigTemplate describing the virtual (emulation) characteristics of the HEmulatedGraphicsDevice .

See Also:
HGraphicsConfigTemplate, HEmulatedGraphicsDevice

getImplementation()

```
public HGraphicsConfigTemplate getImplementation()
```

Returns an HGraphicsConfigTemplate describing the physical (implementation) characteristics of the HEmulatedGraphicsDevice .

Returns:
an HGraphicsConfigTemplate describing the physical (implementation) characteristics of the HEmulatedGraphicsDevice .

See Also:
HGraphicsConfigTemplate, HEmulatedGraphicsDevice

org.havi.ui
HEmulatedGraphicsDevice

Syntax
```
public abstract class HEmulatedGraphicsDevice extends HGraphicsDevice
```

```
java.lang.Object
  |
  +--HScreenDevice
        |
        +--HGraphicsDevice
              |
              +--org.havi.ui.HEmulatedGraphicsDevice
```

All Implemented Interfaces:
org.davic.resources.ResourceProxy, org.davic.resources.ResourceServer

Description
A HEmulatedGraphicsDevice is a "virtual" graphics device that has the capability to be configured to perform one (of many) possible emulations. For example, in the DVB context a 4:3 television might have a HEmulatedGraphicsDevice that had a HEmulatedGraphicsConfiguration that emulated a virtual 14:9 display. The 14:9 HEmulatedGraphicsConfiguration would be used for rendering into from AWT, whilst being displayed on the "true" 4:3 physical display. The relationship between the emulation and implementation is encapsulated within the HEmulatedGraphicsConfiguration .

A HEmulatedGraphicsDevice transforms both AWT pixel-oriented drawing operations and AWT user-input event coordinates, this is performed outside of the Java application (typically in hardware).

A HEmulatedGraphicsDevice may (of necessity) modify coordinates for Components and/or events to the nearest physical / virtual pixel --- authors should not depend on single pixel accuracy.

There is no difference to a Java application between a HGraphicsDevice and a HEmulatedGraphicsDevice , except for the implication of possible rounding errors in integer pixel positions, e.g. Component placement and/or resolution of events.

Java2D mechanisms should behave as per their normal semantics, with respect to display on-screen.

Constructors

HEmulatedGraphicsDevice()
```
protected HEmulatedGraphicsDevice()
```
It is not intended that applications should directly construct HEmulatedGraphicsDevice objects. Creates an HEmulatedGraphicsDevice object. See the class description for details of constructor parameters and default values.

Methods

setGraphicsConfiguration(HEmulatedGraphicsConfiguration)
```
public boolean setGraphicsConfiguration(HEmulatedGraphicsConfiguration hegc)
```

Set the graphics configuration for the device.

Parameters:

`hegc` - the HEmulatedGraphicsConfiguration to which this device should be set.

Returns:

A boolean indicating whether the configuration could be applied successfully. If the configuration could not be applied successfully, the configuration after this method may not match the configuration of the device prior to this method being called --- applications should take steps to determine whether a partial change of settings has been made.

Throws:

`SecurityException` - if the application does not have sufficient rights to set the configuration for this device.

HPermissionDeniedException - (HPermissionDeniedException) if the application does not currently have the right to set the configuration for this device.

HConfigurationException - (HConfigurationException) if the specified configuration is not valid for this device.

org.havi.ui
HFlatEffectMatte

Syntax

```
public class HFlatEffectMatte implements HMatte, HAnimateEffect

java.lang.Object
  |
  +--org.havi.ui.HFlatEffectMatte
```

All Implemented Interfaces:

HAnimateEffect, HMatte

Description

The HFlatEffectMatte class represents a matte that is constant over space but varies over time, it is specified as a sequence of floats. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
data	The transparency value for this flat effect matte.	null (the matte should be treated as being temporally unvarying and opaque)	setMatteData(float[])	getMatteData()

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The intial piece of content to be presented, i.e. its position in the content array.	0	setPosition(int)	getPosition()
By default the effect should be stopped. Hence, to start the effect its start method must be explicitly invoked. This mechanism allows for animations that are programatically controlled, eg via the setPosition method.	"stopped"	start() / stop()	isAnimated()

Constructors

HFlatEffectMatte()

```
public HFlatEffectMatte()
```

Creates an HFlatEffectMatte object. See the class description for details of constructor parameters and default values.

HFlatEffectMatte(float[])

```
public HFlatEffectMatte(float[] data)
```

Creates an HFlatEffectMatte object. See the class description for details of constructor parameters and default values.

Methods

getDelay()

```
public int getDelay()
```

Specified By:
getDelay() in interface HAnimateEffect

See Also:
HAnimateEffect

getMatteData()

```
public float[] getMatteData()
```

Returns the data used for this matte.

Returns:
the data used for this matte.

getPlayMode()

```
public int getPlayMode()
```

Specified By:
getPlayMode() in interface HAnimateEffect

See Also:
HAnimateEffect

getPosition()

```
public int getPosition()
```

Specified By:
getPosition() in interface HAnimateEffect

See Also:
HAnimateEffect

getRepeatCount()

```
public int getRepeatCount()
```

Specified By:
getRepeatCount() in interface HAnimateEffect

See Also:
HAnimateEffect

isAnimated()

```
public boolean isAnimated()
```

Specified By:
isAnimated() in interface HAnimateEffect

See Also:
HAnimateEffect

setDelay(int)

```
public void setDelay(int count)
```

Specified By:
setDelay(int) in interface HAnimateEffect

See Also:
HAnimateEffect

setMatteData(float[])

```
public void setMatteData(float[] data)
```

Sets the data for this matte. Any previously set data is replaced.

Parameters:
`data` - the data for this matte. Specify a null object to remove the associated data for this matte.

setPlayMode(int)

```
public void setPlayMode(int mode)
```

Specified By:
setPlayMode(int) in interface HAnimateEffect

See Also:
HAnimateEffect

setPosition(int)

```
public void setPosition(int position)
```

Specified By:
setPosition(int) in interface HAnimateEffect

See Also:
HAnimateEffect

setRepeatCount(int)

```
public void setRepeatCount(int count)
```

Specified By:
setRepeatCount(int) in interface HAnimateEffect

See Also:
HAnimateEffect

start()

```
public void start()
```

Specified By:
start() in interface HAnimateEffect

See Also:

HAnimateEffect

stop()

```
public void stop()
```

Specified By:
stop() in interface HAnimateEffect

See Also:
HAnimateEffect

org.havi.ui
HFlatMatte

Syntax
```
public class HFlatMatte implements HMatte

java.lang.Object
  |
  +--org.havi.ui.HFlatMatte
```

All Implemented Interfaces:
HMatte

Description
The HFlatMatte class represents a matte that is constant over space and time, it is specified as a float (0.0 is fully transparent and 1.0 fully opaque). The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
data	The transparency value for this flat effect matte.	1.0	setMatteData(float)	getMatteData()

Constructors

HFlatMatte()
```
public HFlatMatte()
```
Creates an HFlatMatte object. See the class description for details of constructor parameters and default values.

HFlatMatte(float)
```
public HFlatMatte(float data)
```
Creates an HFlatMatte object. See the class description for details of constructor parameters and default values.

Methods

getMatteData()
```
public float getMatteData()
```
Returns the data used for this matte.

Returns:

the data used for this matte.

setMatteData(float)

```
public void setMatteData(float data)
```

Sets the data for this matte. Any previously set data is replaced.

Parameters:

`data` - the data for this matte.

org.havi.ui
HFontCapabilities

Syntax

```
public class HFontCapabilities

java.lang.Object
  |
  +--org.havi.ui.HFontCapabilities
```

Description

The HFontCapabilities class allows applications to query the rendering support for various character ranges / individual characters within specified fonts.

Fields

ALPHABETIC_PRESENTATION_FORMS_A

```
public static final int ALPHABETIC_PRESENTATION_FORMS_A
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

ARABIC_EXTENDED

```
public static final int ARABIC_EXTENDED
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

ARABIC_PRESENTATION_FORMS_A

```
public static final int ARABIC_PRESENTATION_FORMS_A
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

ARABIC_PRESENTATION_FORMS_B

```
public static final int ARABIC_PRESENTATION_FORMS_B
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

ARMENIAN

```
public static final int ARMENIAN
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

ARROWS

```
public static final int ARROWS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

BASIC_ARABIC

```
public static final int BASIC_ARABIC
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

BASIC_GEORGIAN

```
public static final int BASIC_GEORGIAN
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

BASIC_GREEK

```
public static final int BASIC_GREEK
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

BASIC_HEBREW

```
public static final int BASIC_HEBREW
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

BASIC_LATIN

```
public static final int BASIC_LATIN
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

BENGALI

```
public static final int BENGALI
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

BI_DIRECTIONAL_FORMAT_EMBEDDINGS

```
public static final int BI_DIRECTIONAL_FORMAT_EMBEDDINGS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

BI_DIRECTIONAL_FORMAT_MARKS

```
public static final int BI_DIRECTIONAL_FORMAT_MARKS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

BLOCK_ELEMENTS

```
public static final int BLOCK_ELEMENTS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

BMP

```
public static final int BMP
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

BOPOMOFO

```
public static final int BOPOMOFO
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

BOX_DRAWING

```
public static final int BOX_DRAWING
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

CHARACTER_SHAPING_SELECTORS

```
public static final int CHARACTER_SHAPING_SELECTORS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

CJK_COMPATIBILITY

```
public static final int CJK_COMPATIBILITY
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

CJK_COMPATIBILITY_FORMS

```
public static final int CJK_COMPATIBILITY_FORMS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

CJK_COMPATIBILITY_IDEOGRAPHS

```
public static final int CJK_COMPATIBILITY_IDEOGRAPHS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

CJK_MISCELLANEOUS

```
public static final int CJK_MISCELLANEOUS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

CJK_SYMBOLS_AND_PUNCTUATION

```
public static final int CJK_SYMBOLS_AND_PUNCTUATION
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

CJK_UNIFIED_IDEOGRAPHS

```
public static final int CJK_UNIFIED_IDEOGRAPHS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

COMBINING_CHARACTERS

```
public static final int COMBINING_CHARACTERS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

COMBINING_CHARACTERS_B_2

```
public static final int COMBINING_CHARACTERS_B_2
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

COMBINING_DIACRITICAL_MARKS

```
public static final int COMBINING_DIACRITICAL_MARKS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

COMBINING_DIACTRICAL_MARKS_FOR_SYMBOLS

```
public static final int COMBINING_DIACTRICAL_MARKS_FOR_SYMBOLS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

COMBINING_HALF_MARKS

```
public static final int COMBINING_HALF_MARKS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

CONTROL_PICTURES

```
public static final int CONTROL_PICTURES
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

CURRENCY_SYMBOLS

```
public static final int CURRENCY_SYMBOLS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

CYRILLIC

```
public static final int CYRILLIC
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

DEVANGARI

```
public static final int DEVANGARI
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

DINGBATS

```
public static final int DINGBATS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

ENCLOSED_ALPHANUMERICS

```
public static final int ENCLOSED_ALPHANUMERICS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

ENCLOSED_CJK_LETTERS_AND_MONTHS

```
public static final int ENCLOSED_CJK_LETTERS_AND_MONTHS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

FORMAT_SEPARATORS

```
public static final int FORMAT_SEPARATORS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

GENERAL_FORMAT_CHARACTERS

```
public static final int GENERAL_FORMAT_CHARACTERS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

GENERAL_PUNCTUATION

```
public static final int GENERAL_PUNCTUATION
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

GEOMETRICAL_SHAPES

```
public static final int GEOMETRICAL_SHAPES
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

GEORGIAN_EXTENDED

```
public static final int GEORGIAN_EXTENDED
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

GREEK_EXTENDED

```
public static final int GREEK_EXTENDED
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

GREEK_SYMBOLS_AND_COPTIC

```
public static final int GREEK_SYMBOLS_AND_COPTIC
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

GUJARATI

```
public static final int GUJARATI
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

GURMUKHI

```
public static final int GURMUKHI
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

HALFWIDTH_AND_FULLWIDTH_FORMS

```
public static final int HALFWIDTH_AND_FULLWIDTH_FORMS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

HANGUL

```
public static final int HANGUL
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

HANGUL_COMPATIBILITY_JAMO

```
public static final int HANGUL_COMPATIBILITY_JAMO
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

HANGUL_FILL_CHARACTERS

```
public static final int HANGUL_FILL_CHARACTERS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

HANGUL_JAMO

```
public static final int HANGUL_JAMO
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

HANGUL_SUPPLEMENTARY_A

```
public static final int HANGUL_SUPPLEMENTARY_A
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

HANGUL_SUPPLEMENTARY_B

```
public static final int HANGUL_SUPPLEMENTARY_B
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

HEBREW_EXTENDED

```
public static final int HEBREW_EXTENDED
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

HIRAGANA

```
public static final int HIRAGANA
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

IPA_EXTENSIONS

```
public static final int IPA_EXTENSIONS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

KANNADA

```
public static final int KANNADA
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

KATAKANA

```
public static final int KATAKANA
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

LAO

```
public static final int LAO
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

LATIN_1_SUPPLEMENT

```
public static final int LATIN_1_SUPPLEMENT
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

LATIN_EXTENDED_A

```
public static final int LATIN_EXTENDED_A
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

LATIN_EXTENDED_ADDITIONAL

```
public static final int LATIN_EXTENDED_ADDITIONAL
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

LATIN_EXTENDED_B

```
public static final int LATIN_EXTENDED_B
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

LETTERLIKE_SYMBOLS

```
public static final int LETTERLIKE_SYMBOLS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

MALAYAM

```
public static final int MALAYAM
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

MATHEMATICAL_OPERATORS

```
public static final int MATHEMATICAL_OPERATORS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

MISCELLANEOUS_SYMBOLS

```
public static final int MISCELLANEOUS_SYMBOLS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

MISCELLANEOUS_TECHNICAL

```
public static final int MISCELLANEOUS_TECHNICAL
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

NUMBER_FORMS

```
public static final int NUMBER_FORMS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

NUMERIC_SHAPE_SELECTORS

```
public static final int NUMERIC_SHAPE_SELECTORS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

OPTICAL_CHARATER_RECOGNITION

```
public static final int OPTICAL_CHARATER_RECOGNITION
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

ORIYA

```
public static final int ORIYA
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

PRIVATE_USE_AREA

```
public static final int PRIVATE_USE_AREA
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

PRIVATE_USE_GROUPS

```
public static final int PRIVATE_USE_GROUPS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

PRIVATE_USE_PLANES

```
public static final int PRIVATE_USE_PLANES
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

SCRIPT_SPECIFIC_FORMAT_CHARACTERS

```
public static final int SCRIPT_SPECIFIC_FORMAT_CHARACTERS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

SMALL_FORM_VARIANTS

```
public static final int SMALL_FORM_VARIANTS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

SPACING_MODIFIER_LETTERS

```
public static final int SPACING_MODIFIER_LETTERS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

SPECIALS

```
public static final int SPECIALS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

SUPERSCRIPTS_AND_SUBSCRIPTS

```
public static final int SUPERSCRIPTS_AND_SUBSCRIPTS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

TAMIL

```
public static final int TAMIL
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

TELUGU

```
public static final int TELUGU
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

THAI

```
public static final int THAI
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

ZERO_WIDTH_BOUNDARY_INDICATORS

```
public static final int ZERO_WIDTH_BOUNDARY_INDICATORS
```

This corresponds to the equivalent character range as defined in ISO/IEC 10646- 1:1993(E) normative Annex A

Constructors

HFontCapabilities()

```
protected HFontCapabilities()
```

It is not intended that applications should directly construct HFontCapabilities objects.

Methods

getSupportedCharacterRanges(Font)

```
public static int[] getSupportedCharacterRanges(java.awt.Font font)
```

Returns the set of character ranges as defined in ISO/IEC 10646-1:1993(E) normative Annex A that this font supports, or a null array if the capabilities of the font are unknown.

Support for a character range does not imply that ALL characters within that range are available in the specified font.

Parameters:

`font` - The font to query for its support for Unicode ranges.

Returns:

An array of integer values, as defined in ISO/IEC 10646-1:1993(E) normative Annex A that this font supports, or null.

isCharAvailable(Font, char)

```
public static boolean isCharAvailable(java.awt.Font font, char c)
```

Returns whether a specific character is available within the specifed font, and can be used as defined in ISO/IEC 10646-1:1993(E) specification by the rendering system, e.g. if rendering of bi-directional text, using BI-DIRECTIONAL_FORMAT_MARKS is supported

Parameters:

`font` - The font to query for its support for the specified character.

`c` - The character whose presence should be tested.

Returns:

true is the character is available within the font and can be rendered as defined in the ISO/IEC 10646- 1:1993(E) specification, false otherwise.

org.havi.ui
HGraphicButton

Syntax

```
public class HGraphicButton extends HIcon implements HActionable, HActionInputPreferred
```

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--HComponent
              |
              +--HVisible
                    |
                    +--HStaticIcon
                          |
                          +--HIcon
                                |
                                +--org.havi.ui.HGraphicButton
```

Direct Known Subclasses:

HToggleButton

All Implemented Interfaces:

HActionable, HActionInputPreferred, HMatteLayer, HNavigable, HNoInputPreferred, HState, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

The HGraphicButton class is used to represent a conventional push-release button used for user actions. This class creates graphical buttons. The different states of the HGraphicButton can have different graphical representations associated with them. By default it uses the HGraphicLook to render itself. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Parameter	Description	Default value	Set method	Get method
width	Width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
height	Height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
image	The image to be used as the content for the NORMAL_STATE , FOCUS_STATE and ACTION_STATE states of this component.	null	setGraphicContent(Image, int)	getGraphicContent(int)
imageNormal	The image to be used as the content for the NORMAL_STATE state of this component.	null	setGraphicContent(Image, int)	getGraphicContent(int)
imagesFocused	The image to be used as the content for the FOCUS_STATE state of this component.	null	setGraphicContent(Image, int)	getGraphicContent(int)
imageActioned	The image to be used as the content for the ACTION_STATE state of this component.	null	setGraphicContent(Image, int)	getGraphicContent(int)

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The default "look" for this class.	A platform specific HGraphicLook	setDefaultLook(HGraphicLook)	getDefaultLook()
The "look" for this object.	The HGraphicLook returned from HGraphicButton.getDefaultLook when this object was created.	setLook(HLook)	getLook()
The gain focus sound.	null	setGainFocusSound(HSound)	getGainFocusSound()
The lose focus sound.	null	setLoseFocusSound(HSound)	getLoseFocusSound()

Description	Default value	Set method	Get method
The action sound.	null	setActionSound(HSound)	getActionSound()

Constructors

HGraphicButton()

```
public HGraphicButton()
```

Creates an HGraphicButton object. See the class description for details of constructor parameters and default values.

HGraphicButton(Image)

```
public HGraphicButton(java.awt.Image image)
```

Creates an HGraphicButton object. See the class description for details of constructor parameters and default values.

HGraphicButton(Image, Image, Image)

```
public HGraphicButton(java.awt.Image imageNormal, java.awt.Image imageFocused, java.awt.
            Image imageActioned)
```

Creates an HGraphicButton object. See the class description for details of constructor parameters and default values.

HGraphicButton(Image, Image, Image, int, int, int, int)

```
public HGraphicButton(java.awt.Image imageNormal, java.awt.Image imageFocused, java.awt.
            Image imageActioned, int x, int y, int width, int height)
```

Creates an HGraphicButton object. See the class description for details of constructor parameters and default values.

HGraphicButton(Image, int, int, int, int)

```
public HGraphicButton(java.awt.Image image, int x, int y, int width, int height)
```

Creates an HGraphicButton object. See the class description for details of constructor parameters and default values.

Methods

addActionListener(ActionListener)

```
public void addActionListener(java.awt.event.ActionListener l)
```

Specified By:
addActionListener(ActionListener) in interface HActionable

See Also:
HActionable

getActionCommand()

```
public java.lang.String getActionCommand()
```

Specified By:
getActionCommand() in interface HActionable

See Also:
HActionable

getActionSound()

```
public HSound getActionSound()
```

Specified By:
getActionSound() in interface HActionable

See Also:
HActionable

getDefaultLook()

```
public static HGraphicLook getDefaultLook()
```

Returns the currently set default HLook for HGraphicButton components.

Returns:
The HLook that is used by default when creating a new HGraphicButton .

getGainFocusSound()

```
public HSound getGainFocusSound()
```

Specified By:
getGainFocusSound() in interface HNavigable

Overrides:
getGainFocusSound() in class HIcon

See Also:
HNavigable

getLoseFocusSound()

```
public HSound getLoseFocusSound()
```

Specified By:
getLoseFocusSound() in interface HNavigable

Overrides:
getLoseFocusSound() in class HIcon

See Also:
HNavigable

getMove(int)

```
public HNavigable getMove(int keyCode)
```

Specified By:
getMove(int) in interface HNavigable

Overrides:
getMove(int) in class HIcon

See Also:
HNavigable

isFocusTraversable()

```
public boolean isFocusTraversable()
```

By default classes implementing HNavigable are focus-traversable.

Overrides:
isFocusTraversable() in class HIcon

Returns:
true

See Also:
`java.awt.Component.isFocusTraversable()`

isSelected()

```
public boolean isSelected()
```

Specified By:
isSelected() in interface HNavigable

Overrides:
isSelected() in class HIcon

See Also:
HNavigable

processActionEvent(ActionEvent)

```
protected void processActionEvent(java.awt.event.ActionEvent evt)
```

Processes an action event enabled for this object. Subclasses of classes implementing processActionEvent should call super.processActionEvent to ensure that the action event is handled appropriately.

Parameters:
`evt` - the event to be processed.

See Also:
HActionable

processEvent(AWTEvent)

```
protected void processEvent(java.awt.AWTEvent evt)
```

It is a valid implementation option for HGraphicButton to override the protected processEvent method.

Overrides:
processEvent(AWTEvent) in class HIcon

See Also:
`java.awt.Component.processEvent(AWTEvent)`

processFocusEvent(FocusEvent)

```
protected void processFocusEvent(java.awt.event.FocusEvent evt)
```

It is a valid implementation option for HGraphicButton to override the protected processFocusEvent method.

Overrides:
processFocusEvent(FocusEvent) in class HIcon

See Also:
`java.awt.Component.processFocusEvent(FocusEvent)`

removeActionListener(ActionListener)

```
public void removeActionListener(java.awt.event.ActionListener l)
```

Specified By:
removeActionListener(ActionListener) in interface HActionable

See Also:
HActionable

setActionCommand(String)

```
public void setActionCommand(java.lang.String command)
```

Specified By:
setActionCommand(String) in interface HActionable

See Also:
HActionable

setActionSound(HSound)

```
public void setActionSound(HSound sound)
```

Specified By:
setActionSound(HSound) in interface HActionable

See Also:
HActionable

setDefaultLook(HGraphicLook)

```
public static void setDefaultLook(HGraphicLook hlook)
```

Sets the default HLook for all HGraphicButton components.

Parameters:
`hlook` - The HLook that will be used by default when creating a new HGraphicButton component.

Throws:
HInvalidLookException - If the HLook is not an HGraphicLook .

setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
            HNavigable right)
```

Specified By:
setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) in interface HNavigable

Overrides:
setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) in class HIcon

See Also:
HNavigable

setGainFocusSound(HSound)

```
public void setGainFocusSound(HSound sound)
```

Specified By:
setGainFocusSound(HSound) in interface HNavigable

Overrides:
setGainFocusSound(HSound) in class HIcon

See Also:
HNavigable

setLoseFocusSound(HSound)

```
public void setLoseFocusSound(HSound sound)
```

Specified By:
setLoseFocusSound(HSound) in interface HNavigable

Overrides:
setLoseFocusSound(HSound) in class HIcon

See Also:
HNavigable

setMove(int, HNavigable)

```
public void setMove(int keyCode, HNavigable target)
```

Specified By:
setMove(int, HNavigable) in interface HNavigable

Overrides:
setMove(int, HNavigable) in class HIcon

See Also:
HNavigable

org.havi.ui
HGraphicLook

Syntax

```
public class HGraphicLook implements HLook

java.lang.Object
  |
  +--org.havi.ui.HGraphicLook
```

All Implemented Interfaces:

java.lang.Cloneable, HLook

Description

The HGraphicLook class is used to getGraphicContent for the current state of the associated HVisible to determine the content to render, and render this on- screen to represent the HVisible . This look will be provided by the platform and the exact way that it is rendered will be platform dependant.

However, the HGraphicLook associated with a given HVisible should use:

- java.awt.Component#getForeground() to determine the Color to render the rectangle.
- Border widths (as set by setHorizontalBorderSpacing / setVerticalBorderSpacing) of 2 pixels by default.

This is the default look that is used by HStaticIcon and its subclasses. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

None.

Default values not exposed in the constructors

Description	Default value	Set method	Get method
Scaling mode	NO_SCALING	setResizeMode(int)	getResizeMode()

See Also:

HStaticIcon

Fields

NO_SCALING

```
public static final int NO_SCALING
```

This value indicates that the look should not attempt to resize the content to fit the widget.

SCALE_ARBITRARY

```
public static final int SCALE_ARBITRARY
```

This value indicates that the look should resize the content to fit the widget. Aspect ratios need not be preserved.

SCALE_PRESERVE

```
public static final int SCALE_PRESERVE
```

This value indicates that the look should resize the content to fit the widget while preserving the aspect ratio of the content. Areas of the widget that are not filled by the content will be look dependent.

Constructors

HGraphicLook()

```
public HGraphicLook()
```

This constructor creates an HGraphicLook .

Methods

getHorizontalBorderSpacing()

```
public int getHorizontalBorderSpacing()
```

Specified By:
getHorizontalBorderSpacing() in interface HLook

See Also:
HLook

getMaximumSize(HVisible)

```
public java.awt.Dimension getMaximumSize(HVisible visible)
```

Specified By:
getMaximumSize(HVisible) in interface HLook

See Also:
HLook

getMinimumSize(HVisible)

```
public java.awt.Dimension getMinimumSize(HVisible visible)
```

Specified By:
getMinimumSize(HVisible) in interface HLook

See Also:
HLook

getPreferredSize(HVisible)

```
public java.awt.Dimension getPreferredSize(HVisible visible)
```

Specified By:
getPreferredSize(HVisible) in interface HLook

See Also:
HLook

getResizeMode()

```
public int getResizeMode()
```

Gets the content resize mode.

Returns:
the current resize mode

getVerticalBorderSpacing()

```
public int getVerticalBorderSpacing()
```

Specified By:
getVerticalBorderSpacing() in interface HLook

See Also:
HLook

setHorizontalBorderSpacing(int)

```
public void setHorizontalBorderSpacing(int width)
```

Specified By:
setHorizontalBorderSpacing(int) in interface HLook

See Also:
HLook

setResizeMode(int)

```
public boolean setResizeMode(int mode)
```

Sets the content resize mode.

Parameters:
`mode` - the desired mode (one of NO_SCALING, SCALE_PRESERVE or SCALE_ARBITRARY)

Returns:
false if the mode is not supported, true otherwise.

setVerticalBorderSpacing(int)

```
public void setVerticalBorderSpacing(int width)
```

Specified By:
setVerticalBorderSpacing(int) in interface HLook

See Also:
HLook

showLook(Graphics, HVisible, int)

```
public void showLook(java.awt.Graphics g, HVisible visible, int state)
```

Specified By:
showLook(Graphics, HVisible, int) in interface HLook

See Also:
HLook

org.havi.ui
HGraphicsConfigTemplate

Syntax

```
public class HGraphicsConfigTemplate extends HScreenConfigTemplate

java.lang.Object
  |
  +--HScreenConfigTemplate
        |
        +--org.havi.ui.HGraphicsConfigTemplate
```

Description

The HGraphicsConfigTemplate class is used to obtain a valid HGraphicsConfiguration . An application instantiates one of these objects and then sets all non-default attributes as desired. The getBestConfiguration(HGraphicsConfigTemplate) method found in the HGraphicsDevice class is then called with this HGraphicsConfigTemplate . A valid HGraphicsConfiguration is returned that meets or exceeds what was requested in the HGraphicsConfigTemplate .

This class this may be subclassed to support define additional properties of backgrounds which may be requested by applications. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

None.

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Fields

IMAGE_SCALING_SUPPORT

```
public static final int IMAGE_SCALING_SUPPORT
```

A value for use in the preference field of the setPreference(int, int) method in the HGraphicsConfigTemplate that indicates that the graphics configuration should or shall support rapid (hardware) image scaling.

MATTE_SUPPORT

```
public static final int MATTE_SUPPORT
```

A value for use in the preference field of the setPreference(int, int) method in the HGraphicsConfigTemplate that indicates that the graphics configuration should or shall support the HAVi mattes feature.

VIDEO_MIXING

```
public static final int VIDEO_MIXING
```

A value for use in the preference field of the setPreference(int, int) method in the HGraphicsConfigTemplate that indicates that the graphics configuration should or shall support transparency in the graphics system such that the output of a video decoder is visible. This includes the following configurations :-

- Configurations where there is a well defined transformation between video pixels and graphics pixels (e.g. pixels are the same size).
- Configurations where an application displays graphics over video but where the video is considered as a background and hence no transformation between the two sets of pixels is required.

Applications may specify a particular video configuration with which mixing must be supported. In this case, the video configuration is specified as an HVideoConfiguration object. If no specific video configuration is required then it is not required to specify such a configuration and null can be used.

Constructors

HGraphicsConfigTemplate()

```
public HGraphicsConfigTemplate()
```

Creates an HGraphicsConfigTemplate object.

Methods

getBestConfiguration(HGraphicsConfigTemplate[])

```
public HGraphicsConfiguration getBestConfiguration(HGraphicsConfigTemplate[] hgcta)
```

The getBestConfiguration method attempts to return an HGraphicsConfiguration that matches the specified HGraphicsConfigTemplate . If this is not possible it will attempt to construct an HEmulatedGraphicsConfiguration where the emulated configuration best matches this HGraphicsConfigTemplate .

Best in this sense means satisfying the preferences in the config template as follows based on the priority (as supplied to HScreenConfigTemplate.setPreference())

1. satisfying all the preferences in that config template whose was REQUIRED
2. excluding configurations with priorites which were REQUIRED_NOT
3. satisfying as many as possible of the preferences whose priority was PREFERRED .
4. Satisfying as few as possible of the preferences whose priority was PREFERRED_NOT .

Parameters:

`hgcta` - - the array of HGraphicsConfigTemplate objects to choose from.

Returns:

an HGraphicsConfiguration object that is the best configuration possible.

org.havi.ui
HGraphicsConfiguration

Syntax

```
public class HGraphicsConfiguration extends HScreenConfiguration

java.lang.Object
  |
  +--HScreenConfiguration
        |
        +--org.havi.ui.HGraphicsConfiguration
```

Direct Known Subclasses:

HEmulatedGraphicsConfiguration

Description

The HGraphicsConfiguration class describes the characteristics (settings) of an HGraphicsDevice. There can be many HGraphicsConfiguration objects associated with a single HGraphicsDevice. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

None.

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

See Also:

HGraphicsDevice

Constructors

HGraphicsConfiguration()

```
protected HGraphicsConfiguration()
```

It is not intended that applications should directly construct HGraphicsConfiguration objects. Creates an HGraphicsConfiguration object. See the class description for details of constructor parameters and default values.

Methods

dispose(Color)

```
public void dispose(java.awt.Color c)
```

This method is used by an application when a colour returned from those versions of the method getPunchThroughToBackgroundColor with a Color as a parameter is no longer equired. It is the responsibility of applications to ensure that no pixels which they had drawn using this colour are still displayed on the screen before calling this method. The result of using such a Color after calling this method is implementation dependent. Using a colour obtained from another source apart from the specified methods will result in this method having no effect.

Parameters:

`c` - the Color which is no longer required

getAllFonts()

```
public java.awt.Font[] getAllFonts()
```

List the fonts that are always available on the device, but does not list fonts that may be (temporarily) available for download from other sources.

getAvailableFontFamilyNames()

```
public java.lang.String[] getAvailableFontFamilyNames()
```

List the font family names that are always available on the device, but does not list font family names that may be (temporarily) available for download from other sources.

getAvailableFontFamilyNames(Locale)

```
public java.lang.String[] getAvailableFontFamilyNames(java.util.Locale l)
```

List the font family names that are always available on the device, but does not list font family names that may be (temporarily) available for download from other sources.

getCompatibleImage(Image, HImageHints)

```
public java.awt.Image getCompatibleImage(java.awt.Image input, HImageHints ih)
```

Modifies a java.awt.Image so that it is compatible with the current HGraphicsConfiguration .

Note: Unmodified Images, or Images modified for other HGraphicsConfiguration should still be able to be rendered within this HGraphicsConfiguration , but may not be as efficient (rapid) in terms of rendering, and may not be presented optimally. For example, an 8bit per RGB component image loaded onto a configuration with a 4bit per RGB component framebuffer may have its pixel values truncated, if this Image is then displayed on an alternate configuration with 16bits per RGB component then it will obviously not be displayed optimally.

The HImageHints provide a mechanism to indicate how any conversion to a constrained graphics environment might best be performed, by describing the general image contents.

It is implementation (and algorithmically) dependent whether this method operates on partial, or complete Image pixel data.

Parameters:

`input` - the java.awt.Image to be modified

`ih` - an HImageHints object that indicates the expected type of the input Image, so that its presentation can be optimally adjusted.

getComponentHScreenRectangle(Component)

```
public HScreenRectangle getComponentHScreenRectangle(java.awt.Component component)
```

Returns the on-screen location of a given visible java.awt.Component as a HScreenRectangle for this HGraphicsDevice .

Parameters:
`component` - the java.awt.Component whose on-screen area is to be determined.

Returns:
the on-screen location of component as a HScreenRectangle for this HGraphicsDevice , or null if the component is not currently added to the HScene (or one of its "child" containers).

See Also:
HScreenRectangle

getConfigTemplate()

```
public HGraphicsConfigTemplate getConfigTemplate()
```

Returns an HGraphicsConfigTemplate object that describes and uniquely identifies this HGraphicsConfiguration .

Hence, the following sequence should return the original HGraphicsConfiguration .

```
HGraphicsDevice.getBestMatch(HGraphicsConfiguration.getConfigTemplate())
```

Properties that are implemented in the HGraphicsConfiguration will return REQUIRED priority, features that are not implemented in the HBackgroundConfiguration will return REQUIRED_NOT priority.

Returns:
an HGraphicsConfigTemplate object which both describes and uniquely identifies this HGraphicsConfiguration .

getDevice()

```
public HGraphicsDevice getDevice()
```

Returns the HGraphicsDevice associated with this HGraphicsConfiguration .

Returns:
the HGraphicsDevice object that is associated with this HGraphicsConfiguration ,

getPixelCoordinatesHScreenRectangle(HScreenRectangle, Container)

```
public java.awt.Rectangle getPixelCoordinatesHScreenRectangle(HScreenRectangle sr, java.
            awt.Container cont)
```

Returns a java.awt.Rectangle which contains the graphics (AWT) pixel area for an HScreenRectangle relative to the supplied java.awt.Container.

Parameters:
`sr` - the screen location expressed as an HScreenRectangle .

`cont` - the java.awt.Container in whose coordinate system the screen location should be expressed.

Returns:
a java.awt.Rectangle which contains the graphics (AWT) pixel area for an HScreenRectangle relative to the supplied java.awt.Container. The returned x, y, width, height values in the java.awt. Rectangle should be such that a

- r = getPixelCoordinatesHScreenRectangle(sr, cont);
- cont.add(component);
- component.setBounds(r.x, r.y, r.width, r.height);

should ensure that the dimensions of the component on-screen should correspond to the given HScreenRectangle , subject to clipping by its parent container, cont.

Note that the HScreenRectangle (HScreenPoint) coordinates are in floats - conversion to pixel coordinate systems necessarily implies a potential loss of precision - however, such conversion should be to the "nearest" integer pixel coordinate.

getPunchThroughToBackgroundColor(Color, int)

```
public java.awt.Color getPunchThroughToBackgroundColor(java.awt.Color color,
            int percentage)
```

This method returns a Color that may be used in standard graphics drawing operations, which has the effect of "punching though" all Components that are behind the Component in which the drawing operation is performed. This includes any visual Components acquired from JMF players. What is behind this HGraphicsConfiguration is revealed through the drawn "hole" blended with the graphics colour specified as the first parameter to this method. Platforms with restricted color spaces may make approximations as required to obtain the best match possible.

Parameters:

`color` - the graphics colour to blend

`percentage` - the blending value for this colour with respect to what is outside this HGraphicsConfiguration . The specified value will be clamped to the range 0 to 100.

Returns:

a Color with the desired effect or null for configurations which do not or are currently unable to support this rendering mode.

getPunchThroughToBackgroundColor(Color, int, HVideoDevice)

```
public java.awt.Color getPunchThroughToBackgroundColor(java.awt.Color color,
            int percentage, HVideoDevice v)
```

This method returns a Color that may be used in standard graphics drawing operations, which has the effect of modifying the existing colour of a pixel to make it partially (or wholly) transparent to the background. The existing pixel percentage transparency to the background at that point shall be equivalent to the (closest) percentage value as specified in the getPunchThroughToBackground-Color percentage parameter.

The existing RGB values of the pixel are unchanged as far as possible, within the limits of the platform. Platforms with restricted color spaces may make approximations as required to obtain the best possible match.

The precise contents of the background are as defined by the platform including any HBackgroundDevice , etc.

Parameters:

`color` - the graphics colour to blend

`percentage` - the alpha value for this colour with respect to what is outside this HGraphicsConfiguration . The specified value will be clamped to the range 0 to 100.

Returns:

a Color with the desired effect or null for configurations which do not or are currently unable to support this rendering mode.

getPunchThroughToBackgroundColor(int)

```
public java.awt.Color getPunchThroughToBackgroundColor(int percentage)
```

This method returns a Color that may be used in standard graphics drawing operations, which has the effect of modifying the existing colour of a pixel to make it partially (or wholly) transparent to the background. The existing pixel percentage transparency to the background at that point shall be equivalent to the (closest) percentage value as specified in the getPunchThroughToBackground-Color percentage parameter.

The existing RGB values of the pixel are unchanged as far as possible, within the limits of the platform. Platforms with restricted color spaces may make approximations as required to obtain the best possible match.

The precise contents of the background are as defined by the platform including any HBackgroundDevice , etc.

Parameters:

`percentage` - the new blending value for each pixel drawn with this colour with respect to what is outside this HGraphicsConfiguration . The specified value will be clamped to the range 0 to 100.

Returns:

a Color with the desired effect or null for configurations which do not or are currently unable to support this rendering mode.

getPunchThroughToBackgroundColor(int, HVideoDevice)

```
public java.awt.Color getPunchThroughToBackgroundColor(int percentage, HVideoDevice hvd)
```

This method returns a Color that may be used in standard graphics drawing operations, which has the effect of "punching though" the HGraphicsDevice in which the drawing operation is performed. The specifed HVideoDevice is revealed through the drawn "hole". The value specified replaces the blending value (with respect to this HVideoDevice) of each pixel drawn with this colour. The existing RGB values of the pixel are unchanged as far as possible within the limits of the platform. Platforms with restricted color spaces may make approximations as required to obtain the best match possible.

Parameters:

`percentage` - the new alpha value for each pixel drawn with this colour with respect to the the HVideoDevice specified. The specified value will be clamped to the range 0 to 100.

`hvd` - the HVideoDevice to reveal.

Returns:

a Color with the desired effect or null for configurations which do not or are currently unable to support this rendering mode.

org.havi.ui
HGraphicsDevice

Syntax
```
public class HGraphicsDevice extends HScreenDevice

java.lang.Object
  |
  +--HScreenDevice
        |
        +--org.havi.ui.HGraphicsDevice
```

Direct Known Subclasses:
HEmulatedGraphicsDevice

All Implemented Interfaces:
org.davic.resources.ResourceProxy, org.davic.resources.ResourceServer

Description
The HGraphicsDevice class describes the raster graphics devices that are available for a particular HScreen. Each HGraphicsDevice has one or more HGraphicsConfiguration objects associated with it. These objects specify the different configurations (settings) in which the HGraphicsDevice can be used.

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

See Also:
HGraphicsConfiguration

Constructors

HGraphicsDevice()
```
protected HGraphicsDevice()
```
It is not intended that applications should directly construct HGraphicsDevice objects. Creates an HGraphicsDevice object. See the class description for details of constructor parameters and default values.

Methods

getBestConfiguration(HGraphicsConfigTemplate)
```
public HGraphicsConfiguration getBestConfiguration(HGraphicsConfigTemplate hgct)
```
The getBestConfiguration method attempts to return an HGraphicsConfiguration that matches the specified HGraphicsConfigTemplate. If this is not possible it will attempt to construct an

HEmulatedGraphicsConfiguration where the emulated configuration best matches this HGraphicsConfigTemplate .

Best in this sense means satisfying the preferences in the config template as follows based on the priority (as supplied to HScreenConfigTemplate.setPreference())

1. satisfying all the preferences in that config template whose was REQUIRED
2. excluding configurations with priorites which were REQUIRED_NOT
3. satisfying as many as possible of the preferences whose priority was PREFERRED .
4. Satisfying as few as possible of the preferences whose priority was PREFERRED_NOT .

Parameters:
`hgct` - an HGraphicsConfigTemplate object used to obtain a valid HGraphicsConfiguration .

Returns:
an HGraphicsConfiguration that passes the criteria defined in the specified HGraphicsConfigTemplate ,

getBestConfiguration(HGraphicsConfigTemplate[])

```
public HGraphicsConfiguration getBestConfiguration(HGraphicsConfigTemplate[] hgcta)
```

The getBestConfiguration method attempts to return an HGraphicsConfiguration that matches the specified HGraphicsConfigTemplate objects within the specified array. The HGraphicsTemplate objects should be considered for matching in priority order from 0 to (hgcta.length - 1). If this is not possible, it will attempt to construct an HEmulatedGraphicsConfiguration where the emulated configuration best matches the specified HGraphicsConfigTemplate objects.

Best in this sense means satisfying the preferences in the config template as follows based on the priority (as supplied to HScreenConfigTemplate.setPreference())

1. satisfying all the preferences in that config template whose was REQUIRED
2. excluding configurations with priorites which were REQUIRED_NOT
3. satisfying as many as possible of the preferences whose priority was PREFERRED .
4. Satisfying as few as possible of the preferences whose priority was PREFERRED_NOT .

Parameters:
`hgcta` - the HGraphicsConfigTemplate array used to obtain a valid HGraphicsConfiguration .

Returns:
an HGraphicsConfiguration that passes the criteria defined in one of the HGraphicsConfigTemplate objects within the specified array. The class of the object returned can also be a class inheriting from HGraphicsConfiguration (e.g. HEmulatedGraphicsConfiguration).

getConfigurations()

```
public HGraphicsConfiguration[] getConfigurations()
```

Returns all of the HGraphicsConfiguration objects associated with this HGraphicsDevice .

Returns:
an array of HGraphicsConfiguration objects that are associated with this HGraphicsDevice . The class of the objects returned can also be a class inheriting from HGraphicsConfiguration (e.g. HEmulatedGraphicsConfiguration).

See Also:
HGraphicsConfiguration

getCurrentConfiguration()

```
public HGraphicsConfiguration getCurrentConfiguration()
```

Returns the current HGraphicsConfiguration for this HGraphicsDevice .

Returns:
the current HGraphicsConfiguration for this HGraphicsDevice . The class of the object returned can also be a class inheriting from HGraphicsConfiguration (e.g. HEmulatedGraphicsConfiguration).

See Also:
HGraphicsConfiguration

getDefaultConfiguration()

```
public HGraphicsConfiguration getDefaultConfiguration()
```

Returns the default HGraphicsConfiguration associated with this HGraphicsDevice . This (single) default configuration should correspond to some well-behaved settings for the device, such as, a minimal configuration, or factory preset settings.

Returns:
the default HGraphicsConfiguration of this HGraphicsDevice . The class of the object returned can also be a class inheriting from HGraphicsConfiguration (e.g. HEmulatedGraphicsConfiguration).

setGraphicsConfiguration(HGraphicsConfiguration)

```
public boolean setGraphicsConfiguration(HGraphicsConfiguration hgc)
```

Set the graphics configuration for the device.

Parameters:
`hgc` - the HGraphicsConfiguration to which this device should be set.

Returns:
A boolean indicating whether the configuration could be applied successfully. If the configuration could not be applied successfully, the configuration after this method may not match the configuration of the device prior to this method being called --- applications should take steps to determine whether a partial change of settings has been made.

Throws:
`SecurityException` - if the application does not have sufficient rights to set the configuration for this device.

HPermissionDeniedException - (HPermissionDeniedException) if the application does not currently have the right to set the configuration for this device.

HConfigurationException - (HConfigurationException) if the specified configuration is not valid for this device.

org.havi.ui
HIcon

Syntax

```
public class HIcon extends HStaticIcon implements HNavigable
```

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--HComponent
              |
              +--HVisible
                    |
                    +--HStaticIcon
                          |
                          +--org.havi.ui.HIcon
```

Direct Known Subclasses:

HGraphicButton

All Implemented Interfaces:

HMatteLayer, HNavigable, HNoInputPreferred, HState, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

The HIcon class creates an a graphical image. By default it uses the HGraphicLook to render itself. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	Width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Parameter	Description	Default value	Set method	Get method
height	Height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
image	The image to be used as the content for the NORMAL_STATE , FOCUS_STATE and ACTION_STATE states of this component.	null	setGraphicContent(Image, int)	getGraphicContent(int)

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The default "look" for this class.	A platform specific HGraphicLook	setDefaultLook(HGraphicLook)	getDefaultLook()
The "look" for this object.	The HGraphicLook returned from HIcon.getDefaultLook when this object was created.	setLook(HLook)	getLook()
The gain focus sound.	null	setGainFocusSound(HSound)	getGainFocusSound()
The lose focus sound.	null	setLoseFocusSound(HSound)	getLoseFocusSound()

Constructors

HIcon()

```
public HIcon()
```

Creates an HIcon object. See the class description for details of constructor parameters and default values.

HIcon(Image)

```
public HIcon(java.awt.Image image)
```

Creates an HIcon object. See the class description for details of constructor parameters and default values.

HIcon(Image, int, int, int, int)

```
public HIcon(java.awt.Image image, int x, int y, int width, int height)
```

Creates an HIcon object. See the class description for details of constructor parameters and default values.

Methods

getDefaultLook()

```
public static HGraphicLook getDefaultLook()
```

Returns the currently set default look for HIcon components.

Returns:
The look that is used by default when creating a new HIcon component.

getGainFocusSound()

```
public HSound getGainFocusSound()
```

Specified By:
getGainFocusSound() in interface HNavigable

See Also:
HNavigable

getLoseFocusSound()

```
public HSound getLoseFocusSound()
```

Specified By:
getLoseFocusSound() in interface HNavigable

See Also:
HNavigable

getMove(int)

```
public HNavigable getMove(int keyCode)
```

Specified By:
getMove(int) in interface HNavigable

See Also:
HNavigable

isFocusTraversable()

```
public boolean isFocusTraversable()
```

By default classes implementing HNavigable are focus-traversable.

Overrides:
isFocusTraversable() in class HVisible

Returns:
true

See Also:
`java.awt.Component.isFocusTraversable()`

isSelected()

```
public boolean isSelected()
```

Specified By:
isSelected() in interface HNavigable

See Also:
HNavigable

processEvent(AWTEvent)

```
protected void processEvent(java.awt.AWTEvent evt)
```

It is a valid implementation option for HIcon to override the protected processEvent method.

Overrides:
java.awt.Component.processEvent(java.awt.AWTEvent) in class java.awt.Component

See Also:
`java.awt.Component.processEvent(AWTEvent)`

processFocusEvent(FocusEvent)

```
protected void processFocusEvent(java.awt.event.FocusEvent evt)
```

It is a valid implementation option for HIcon to override the protected processFocusEvent method.

Overrides:
java.awt.Component.processFocusEvent(java.awt.event.FocusEvent) in class java.awt.Component

See Also:
`java.awt.Component.processFocusEvent(FocusEvent)`

requestFocus()

```
public void requestFocus()
```

Specified By:
requestFocus() in interface HNavigable

Overrides:
java.awt.Component.requestFocus() in class java.awt.Component

See Also:
`java.awt.Component.requestFocus()`

setDefaultLook(HGraphicLook)

```
public static void setDefaultLook(HGraphicLook hlook)
```

Sets the default HLook for all HIcon Components.

Parameters:
`hlook` - The HLook that will be used by default when creating a new HIcon component.

Throws:
HInvalidLookException - If the HLook is not an HGraphicLook .

setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
            HNavigable right)
```

Specified By:
setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) in interface HNavigable

See Also:
HNavigable

setGainFocusSound(HSound)

```
public void setGainFocusSound(HSound sound)
```

Specified By:
setGainFocusSound(HSound) in interface HNavigable

See Also:
HNavigable

setLoseFocusSound(HSound)

```
public void setLoseFocusSound(HSound sound)
```

Specified By:
setLoseFocusSound(HSound) in interface HNavigable

See Also:
HNavigable

setMove(int, HNavigable)

```
public void setMove(int keyCode, HNavigable target)
```

Specified By:
setMove(int, HNavigable) in interface HNavigable

See Also:
HNavigable

org.havi.ui
HImageEffectMatte

Syntax

```
public class HImageEffectMatte implements HMatte, HAnimateEffect

java.lang.Object
  |
  +--org.havi.ui.HImageEffectMatte
```

All Implemented Interfaces:

HAnimateEffect, HMatte

Description

The HImageEffectMatte class represents a matte that varies over both space and time, it is specified as a sequence of image masks. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
data	The transparency value for this flat effect matte.	null (the matte should be treated as being spatially and temporally unvarying and opaque)	setMatteData(Image[])	getMatteData()

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The intial piece of content to be presented, i.e. its position in the content array.	0	setPosition(int)	getPosition()
By default the effect should be stopped. Hence, to start the effect its start method must be explicitly invoked. This mechanism allows for animations that are programatically controlled, eg via the setPosition method.	"stopped"	start() / stop()	isAnimated()
The pixel offset for each image within the HImageEffectMatte , relative to the top, left corner of its associated component.	A java.awt.Point = (0, 0)	setOffset(Point, int)	getOffset(int)

Constructors

HImageEffectMatte()

```
public HImageEffectMatte()
```

Creates an HImageEffectMatte object. See the class description for details of constructor parameters and default values.

HImageEffectMatte(Image[])

```
public HImageEffectMatte(java.awt.Image[] data)
```

Creates an HImageEffectMatte object. See the class description for details of constructor parameters and default values.

Methods

getDelay()

```
public int getDelay()
```

Specified By:
getDelay() in interface HAnimateEffect

See Also:
HAnimateEffect

getMatteData()

```
public java.awt.Image[] getMatteData()
```

Returns the data used for this matte.

Returns:
the data used for this matte.

getOffset(int)

```
public java.awt.Point getOffset(int index)
```

Get the offset of a specified frame of the matte relative to its component in pixels.

Parameters:
`index` - the zero-index to the data for which the offset should be recovered.

Returns:
the offset of the specified frame of the matte relative to its component in pixels (as a Point)

getPlayMode()

```
public int getPlayMode()
```

Specified By:
getPlayMode() in interface HAnimateEffect

See Also:
HAnimateEffect

getPosition()

```
public int getPosition()
```

Specified By:
getPosition() in interface HAnimateEffect

See Also:
HAnimateEffect

getRepeatCount()

```
public int getRepeatCount()
```

Specified By:
getRepeatCount() in interface HAnimateEffect

See Also:
HAnimateEffect

isAnimated()

```
public boolean isAnimated()
```

Specified By:
isAnimated() in interface HAnimateEffect

See Also:
HAnimateEffect

setDelay(int)

```
public void setDelay(int count)
```

Specified By:
setDelay(int) in interface HAnimateEffect

See Also:
HAnimateEffect

setMatteData(Image[])

```
public void setMatteData(java.awt.Image[] data)
```

Sets the data for this matte. Any previously set data is replaced.

Parameters:
`data` - the data for this matte. Specify a null object to remove the associated data for this matte.

setOffset(Point, int)

```
public void setOffset(java.awt.Point p, int index)
```

setPlayMode(int)

```
public void setPlayMode(int mode)
```

Specified By:
setPlayMode(int) in interface HAnimateEffect

See Also:
HAnimateEffect

setPosition(int)

```
public void setPosition(int position)
```

Specified By:
setPosition(int) in interface HAnimateEffect

See Also:
HAnimateEffect

setRepeatCount(int)

```
public void setRepeatCount(int count)
```

Specified By:
setRepeatCount(int) in interface HAnimateEffect

See Also:
HAnimateEffect

start()

```
public void start()
```

Specified By:
start() in interface HAnimateEffect

See Also:
HAnimateEffect

stop()

```
public void stop()
```

Specified By:
stop() in interface HAnimateEffect

See Also:
HAnimateEffect

org.havi.ui
HImageHints

Syntax

```
public class HImageHints

java.lang.Object
  |
  +--org.havi.ui.HImageHints
```

Description

The HImageHints object allows an application to pass hints to the system how best to tailor an image to match a (possibly) restricted HGraphicsConfiguration . The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

None.

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The default image hint.	NATURAL_IMAGE		

Fields

BUSINESS_GRAPHICS

```
public static final int BUSINESS_GRAPHICS
```

The image is business graphics, with strong, well-defined, blocks of solid colour, etc. Not-suitable for dithering, suitable for nearest colour matching

CARTOON

```
public static final int CARTOON
```

The image is a cartoon, with strong, well-defined, blocks of solid colour, etc. Not-suitable for dithering, suitable for nearest colour matching

LINE_ART

```
public static final int LINE_ART
```

The image is a two-tone lineart, with colours varying between foreground and background, etc. Not-suitable for dithering. Possibly suitable for colour-map adjustment, etc., if applicable.

NATURAL_IMAGE

```
public static final int NATURAL_IMAGE
```

The image is a "natural" scene, with subtle gradations of colour, etc. Suitable for dithering.

Constructors

HImageHints()

```
public HImageHints()
```

Methods

getType()

```
public int getType()
```

Get the expected type of the image being loaded.

setType(int)

```
public void setType(int type)
```

Set the expected type of the image being loaded.

org.havi.ui
HImageMatte

Syntax

```
public class HImageMatte implements HMatte

java.lang.Object
  |
  +--org.havi.ui.HImageMatte
```

All Implemented Interfaces:

HMatte

Description

The HImageMatte class represents a matte that varies over space but is constant over time, it can be specified by an "image mask" (a single channel image) where the pixels indicate matte transparency. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
data	The transparency value for this image effect matte.	null (the matte should be treated as being spatially unvarying and opaque)	setMatteData(Image)	getMatteData()

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The pixel offset for the image matte, relative to the top, left corner of its associated component.	A java.awt.Point = (0, 0)	setOffset(Point)	getOffset()

Constructors

HImageMatte()

```
public HImageMatte()
```

Creates an HImageMatte object. See the class description for details of constructor parameters and default values.

HImageMatte(Image)

```
public HImageMatte(java.awt.Image data)
```

Creates an HImageMatte object. See the class description for details of constructor parameters and default values.

Methods

getMatteData()

```
public java.awt.Image getMatteData()
```

Returns the data used for this matte.

Returns:
the data used for this matte.

getOffset()

```
public java.awt.Point getOffset()
```

Get the offset of the matte relative to its component in pixels.

Returns:
the offset of the specified frame of the matte relative to its component in pixels (as a Point)

setMatteData(Image)

```
public void setMatteData(java.awt.Image data)
```

Sets the data for this matte. Any previously set data is replaced.

Parameters:
`data` - the data for this matte. Specify a null object to remove the associated data for this matte.

setOffset(Point)

```
public void setOffset(java.awt.Point p)
```

Set the offset of the matte relative to its component in pixels.

Parameters:
`p` - the offset of the matte relative to its component in pixels

org.havi.ui
HInvalidLookException

Syntax

```
public class HInvalidLookException extends HUIException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--HUIException
                    |
                    +--org.havi.ui.HInvalidLookException
```

All Implemented Interfaces:

java.io.Serializable

Description

A HInvalidLookException is an exception that is thrown when a a particular look is not compatable with the widget it has been associated with. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Constructors

HInvalidLookException()

```
public HInvalidLookException()
```

org.havi.ui
HKeyboardInputPreferred

Syntax
```
public interface HKeyboardInputPreferred
```

All Known Implementing Classes:
HSinglelineEntry

Description
A component which implements HKeyboardInputPreferred indicates that alphanumeric entry is required.

Platforms without keyboards will provide another means for keyboard entry, when this component gains focus, for example, by offering an on-screen keyboard.

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

org.havi.ui
HListElement

Syntax

```
public class HListElement extends HVisible implements HSwitchable, HActionInputPreferred

java.lang.Object
  |
  +--java.awt.Component
        |
        +--HComponent
              |
              +--HVisible
                    |
                    +--org.havi.ui.HListElement
```

All Implemented Interfaces:

HActionable, HActionInputPreferred, HMatteLayer, HNavigable, HState, HSwitchable, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

The HListElement class is an HSwitchable component and hence maintains a switchable state as to whether it is currently "selected" or not, as would be expected in a list type control. An HListElement is **only** intended to be used in conjunction with an HListGroup which manages its layout, focus, etc., to provide list functionality. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

None.

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The default "look" for this class.	A platform specific HTextLook	setDefaultLook(HLook)	getDefaultLook()
The "look" for this object.	The HTextLook , or HGraphicLook returned from HListElement.getDefaultLook when this object was created.	setLook(HLook)	getLook()
The gain focus sound.	null	setGainFocusSound(HSound)	getGainFocusSound()
The lose focus sound.	null	setLoseFocusSound(HSound)	getLoseFocusSound()
The action sound.	null	setActionSound(HSound)	getActionSound()
The unset action sound.	null	setUnsetActionSound(HSound)	getUnsetActionSound()

Constructors

HListElement()

```
public HListElement()
```

Creates a HListElement widget that uses the default look returned from the method getDefaultLook.

HListElement(Image, HGraphicLook)

```
public HListElement(java.awt.Image image, HGraphicLook graphicLook)
```

Creates a HListElement widget that uses the look specified in the parameters, with the content specified in the parameters.

Parameters:

`image` - the graphical content to be used for this HListElement

`graphicLook` - the HGraphicLook to be used for this HListElement

HListElement(String, HTextLook)

```
public HListElement(java.lang.String text, HTextLook textLook)
```

Creates a HListElement widget that uses the look specified in the parameters, with the content specified in the parameters.

Parameters:

`text` - the textual content to be used for this HListElement

`textLook` - the HTextLook to be used for this HListElement

Methods

addActionListener(ActionListener)

```
public void addActionListener(java.awt.event.ActionListener l)
```

Specified By:

addActionListener(ActionListener) in interface HActionable

See Also:

HActionable

getActionCommand()

```
public java.lang.String getActionCommand()
```

Specified By:

getActionCommand() in interface HActionable

See Also:

HActionable

getActionSound()

```
public HSound getActionSound()
```

Specified By:

getActionSound() in interface HActionable

See Also:

HActionable

getDefaultLook()

```
public static HLook getDefaultLook()
```

Returns the currently set default HLook for HListElement components.

Returns:
The HLook that is used by default when creating a new HListElement component.

getGainFocusSound()

```
public HSound getGainFocusSound()
```

Specified By:
getGainFocusSound() in interface HNavigable

See Also:
HNavigable

getLoseFocusSound()

```
public HSound getLoseFocusSound()
```

Specified By:
getLoseFocusSound() in interface HNavigable

See Also:
HNavigable

getMove(int)

```
public HNavigable getMove(int keyCode)
```

Specified By:
getMove(int) in interface HNavigable

See Also:
HNavigable

getSwitchableState()

```
public boolean getSwitchableState()
```

Specified By:
getSwitchableState() in interface HSwitchable

See Also:
HSwitchable

getUnsetActionSound()

```
public HSound getUnsetActionSound()
```

Specified By:
getUnsetActionSound() in interface HSwitchable

See Also:
HSwitchable

isFocusTraversable()

```
public boolean isFocusTraversable()
```

By default classes implementing HNavigable are focus-traversable.

Overrides:
isFocusTraversable() in class HVisible

Returns:
true

See Also:
`java.awt.Component.isFocusTraversable()`

isSelected()

```
public boolean isSelected()
```

Specified By:
isSelected() in interface HNavigable

See Also:
HNavigable

processActionEvent(ActionEvent)

```
protected void processActionEvent(java.awt.event.ActionEvent evt)
```

Processes an action event enabled for this object. Subclasses of classes implementing processActionEvent should call super.processActionEvent to ensure that the action event is handled appropriately.

Parameters:
`evt` - the event to be processed.

See Also:
HActionable

processEvent(AWTEvent)

```
protected void processEvent(java.awt.AWTEvent evt)
```

It is a valid implementation option for HListElement to override the protected processEvent method.

Overrides:
java.awt.Component.processEvent(java.awt.AWTEvent) in class java.awt.Component

See Also:
`java.awt.Component.processEvent(AWTEvent)`

processFocusEvent(FocusEvent)

```
protected void processFocusEvent(java.awt.event.FocusEvent evt)
```

It is a valid implementation option for HListElement to override the protected processFocusEvent method.

Overrides:
java.awt.Component.processFocusEvent(java.awt.event.FocusEvent) in class java.awt.Component

See Also:
`java.awt.Component.processFocusEvent(FocusEvent)`

removeActionListener(ActionListener)

```
public void removeActionListener(java.awt.event.ActionListener l)
```

Specified By:
removeActionListener(ActionListener) in interface HActionable

See Also:
HActionable

requestFocus()

```
public void requestFocus()
```

Specified By:
requestFocus() in interface HNavigable

Overrides:
java.awt.Component.requestFocus() in class java.awt.Component

See Also:
`java.awt.Component.requestFocus()`

setActionCommand(String)

```
public void setActionCommand(java.lang.String command)
```

Specified By:
setActionCommand(String) in interface HActionable

See Also:
HActionable

setActionSound(HSound)

```
public void setActionSound(HSound sound)
```

Specified By:
setActionSound(HSound) in interface HActionable

See Also:
HActionable

setDefaultLook(HLook)

```
public static void setDefaultLook(HLook hlook)
```

Sets the default HLook for all HListElement components.

Parameters:
`hlook` - The HLook that will be used by default when creating a new HListElement .

Throws:
HInvalidLookException - If the HLook is not an HTextLook , or an HGraphicLook .

setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
            HNavigable right)
```

Specified By:
setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) in interface HNavigable

See Also:
HNavigable

setGainFocusSound(HSound)

```
public void setGainFocusSound(HSound sound)
```

Specified By:
setGainFocusSound(HSound) in interface HNavigable

See Also:
HNavigable

setLook(HLook)

```
public void setLook(HLook hlook)
```

Sets the HLook for this component.

Overrides:
setLook(HLook) in class HVisible

Parameters:
`hlook` - The HLook that is to be used for this component.

Throws:
HInvalidLookException - If the HLook is not an HTextLook , or an HGraphicLook .

setLoseFocusSound(HSound)

```
public void setLoseFocusSound(HSound sound)
```

Specified By:
setLoseFocusSound(HSound) in interface HNavigable

See Also:
HNavigable

setMove(int, HNavigable)

```
public void setMove(int keyCode, HNavigable target)
```

Specified By:
setMove(int, HNavigable) in interface HNavigable

See Also:
HNavigable

setSwitchableState(boolean)

```
public void setSwitchableState(boolean state)
```

Specified By:
setSwitchableState(boolean) in interface HSwitchable

See Also:
HSwitchable

setUnsetActionSound(HSound)

```
public void setUnsetActionSound(HSound sound)
```

Specified By:
setUnsetActionSound(HSound) in interface HSwitchable

See Also:
HSwitchable

org.havi.ui
HListGroup

Syntax

```
public class HListGroup extends HContainer implements HState, HValue, HAdjustmentInputPreferred
```

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--HContainer
                    |
                    +--org.havi.ui.HListGroup
```

All Implemented Interfaces:

HAdjustmentInputPreferred, HMatteLayer, HNavigable, HState, HValue, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

A HListGroup manages a set of HListElement and presents these elements to represent a scrollable list. The HListGroup manages which HListElement will appear on screen. That is the HListGroup manages the scrolling of the HListElement and manages the focus navigation between the HListElement . The HListGroup is Navigable, but when focus is received by the HListGroup object it will transfer focus to the appropriate HListElement within the group. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	Width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
height	Height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Parameter	Description	Default value	Set method	Get method
items	The initial list of elements	null	add(Component)	---
numVisible	A hint as to the number of HListElement that should be visible within the HListGroup .	0	setNumVisibleElements(int)	getNumVisibleElements()

Default values not exposed in the constructors

Description	Default value	Set method	Get method
HListGroup default LayoutManager	HListGroupLayoutManager	java.awt.Container#getLayout	java.awt.Container#setLayout
The ability of an HListGroup to perform multiple selections.	false s	setMultiSelection(boolean)	getMultiSelection()
An added HRange component, indicating position within the HListGroup	none added	addScrollbar(HRange)	---

The HListGroup is an HContainer and the default layout manager used by the HListGroup is a HListGroupLayoutManager with a VERTICAL alignment.

See Also:

HListGroupLayoutManager

Constructors

HListGroup()

```
public HListGroup()
```

Creates an HListGroup object. See the class description for details of constructor parameters and default values.

HListGroup(HListElement[], int, int, int, int, int)

```
public HListGroup(HListElement[] items, int numVisible, int x, int y, int width,
            int height)
```

Creates an HListGroup object. See the class description for details of constructor parameters and default values.

Methods

add(Component)

```
public java.awt.Component add(java.awt.Component comp)
```

Adds the specified component to the end of the HListGroup . Only HListElement can be added to a HListGroup , adding other types of Components will fail and return null.

Overrides:
java.awt.Container.add(java.awt.Component) in class java.awt.Container

Parameters:
`comp` - the component to be added.

Returns:
the component if it was successfully added to the HListGroup , otherwise null.

add(Component, int)

```
public java.awt.Component add(java.awt.Component comp, int index)
```

Adds the specified component to this HListGroup at the given position. Only HListElement can be added to a HListGroup , adding other types of Components will fail and return null.

Overrides:
java.awt.Container.add(java.awt.Component, int) in class java.awt.Container

Parameters:
`comp` - the component to be added.

`index` - the position at which to insert the component, or -1 to insert the component at the end.

Returns:
the component if it was successfully added to the HListGroup , otherwise null.

add(Component, Object)

```
public void add(java.awt.Component comp, java.lang.Object constraints)
```

Adds the specified component to the end of the HListGroup . Only HListElement can be added to a HListGroup , adding other types of Components will fail. Also notifies the layout manager to add the component to this container's layout using the specified constraints object.

Overrides:
java.awt.Container.add(java.awt.Component, java.lang.Object) in class java.awt.Container

Parameters:
`comp` - the component to be added.

`constraints` - an object expressing layout constraints for this component

add(Component, Object, int)

```
public void add(java.awt.Component comp, java.lang.Object constraints, int index)
```

Adds the specified component to this HListGroup at the given position. Only HListElement can be added to a HListGroup , adding other types of Components will fail. Also notifies the layout manager to add the component to this container's layout using the specified constraints object.

Overrides:
java.awt.Container.add(java.awt.Component, java.lang.Object, int) in class java.awt.Container

Parameters:
`comp` - the component to be added.

`constraints` - an object expressing layout constraints for this component

`index` - the position at which to insert the component, or -1 to insert the component at the end.

add(HListElement[])

```
public void add(HListElement[] elements)
```

Adds an array of HListElement in order, at the end of the HListGroup .

Parameters:

`elements` - The array of HListElement to be added to the HListGroup .

add(HListElement[], int)

```
public void add(HListElement[] elements, int index)
```

Adds an array of HListElement in order at the given position.

Parameters:

`elements` - The array of HListElement to be added to the HListGroup .

`index` - the position at which to insert the HListElement , or -1 to insert the elements at the end.

Returns:

the component if it was successfully added to the HListGroup , otherwise null.

add(String, Component)

```
public java.awt.Component add(java.lang.String name, java.awt.Component comp)
```

Adds the specified component to this container. It is strongly advised to use the 1.1 method, add(Component, Object), in place of this method.

Overrides:

java.awt.Container.add(java.lang.String, java.awt.Component) in class java.awt.Container

addChangeListener(HValueChangeListener)

```
public void addChangeListener(HValueChangeListener l)
```

Specified By:

addChangeListener(HValueChangeListener) in interface HValue

See Also:

HValue

addScrollbar(HRange)

```
public void addScrollbar(HRange bar)
```

Associates an HRange with the list. The HListGroup will adjust the HRange to indicate the position currently in the list. Note the HListGroup has no control over the positioning or look of the HRange control, which must be setup appropriately by the application author.

getChangeSound()

```
public HSound getChangeSound()
```

Specified By:

getChangeSound() in interface HValue

See Also:

HValue

getCurrentElement()

```
public int getCurrentElement()
```

Get the zero-based index of the currently focussed HListElement within the HListGroup .

Returns:
the zero-based index of the HListElement that has focus within the HListGroup .

getFirstVisible()

```
public int getFirstVisible()
```

Get the zero-based index of the first HListElement that is visible within the HListGroup .

Returns:
the zero-based index of the first HListElement that is visible within the HListGroup .

getGainFocusSound()

```
public HSound getGainFocusSound()
```

Specified By:
getGainFocusSound() in interface HNavigable

See Also:
HNavigable

getLoseFocusSound()

```
public HSound getLoseFocusSound()
```

Specified By:
getLoseFocusSound() in interface HNavigable

See Also:
HNavigable

getMove(int)

```
public HNavigable getMove(int keyCode)
```

Specified By:
getMove(int) in interface HNavigable

See Also:
HNavigable

getMultiSelection()

```
public boolean getMultiSelection()
```

Indicates if the list is currently set for single or multiple selection.

Returns:
true if the list is a multi-selection list, otherwise returns false for single selection.

getNumberSelected()

```
public int getNumberSelected()
```

Returns the number of HListElement selected.

getNumVisibleElements()

```
public int getNumVisibleElements()
```

Gets the number of visible elements in the HListGroup .

Returns:
Number of visible elements.

getSelection()

```
public HListElement[] getSelection()
```

Returns an array of HListElement that have been selected from the list.

If the list is a single selection HListGroup the array will contain a maximum of one HListElement .

isFocusTraversable()

```
public boolean isFocusTraversable()
```

By default classes implementing HNavigable are focus-traversable.

Overrides:
java.awt.Component.isFocusTraversable() in class java.awt.Component

Returns:
true

See Also:
`java.awt.Component.isFocusTraversable()`

isSelected()

```
public boolean isSelected()
```

Specified By:
isSelected() in interface HNavigable

See Also:
HNavigable

processChangeEvent(HValueChangeEvent)

```
protected void processChangeEvent(HValueChangeEvent evt)
```

Processes HValueChangeEvent enabled for this object. Subclasses of classes implementing processChangeEvent should call super.processChangeEvent to ensure that the change event is handled appropriately.

Parameters:
`evt` - the event to be processed.

See Also:
HValue

processEvent(AWTEvent)

```
protected void processEvent(java.awt.AWTEvent evt)
```

It is a valid implementation option for HListGroup to override the protected processEvent method.

Overrides:
java.awt.Container.processEvent(java.awt.AWTEvent) in class java.awt.Container

See Also:
`java.awt.Component.processEvent(AWTEvent)`

processFocusEvent(FocusEvent)

```
protected void processFocusEvent(java.awt.event.FocusEvent evt)
```

It is a valid implementation option for HListGroup to override the protected processFocusEvent method.

Overrides:

java.awt.Component.processFocusEvent(java.awt.event.FocusEvent) in class java.awt.Component

See Also:
`java.awt.Component.processFocusEvent(FocusEvent)`

processKeyEvent(KeyEvent)

```
protected void processKeyEvent(java.awt.event.KeyEvent evt)
```

It is a valid implementation option for HListGroup to override the protected processKeyEvent method.

Overrides:
java.awt.Component.processKeyEvent(java.awt.event.KeyEvent) in class java.awt.Component

See Also:
`java.awt.Component.processKeyEvent(KeyEvent)`

removeChangeListener(HValueChangeListener)

```
public void removeChangeListener(HValueChangeListener l)
```

Specified By:
removeChangeListener(HValueChangeListener) in interface HValue

See Also:
HValue

setChangeSound(HSound)

```
public void setChangeSound(HSound sound)
```

Specified By:
setChangeSound(HSound) in interface HValue

See Also:
HValue

setCurrentElement(int)

```
public void setCurrentElement(int current)
```

Set the zero-based index of the currently focussed HListElement within the HListGroup .

Parameters:
`current` - the zero-based index of the HListElement that should have focus within the HListGroup .

setFirstVisible(int)

```
public void setFirstVisible(int first)
```

Set the zero-based index of the first HListElement that should be visible within the HListGroup .

Parameters:
`first` - the first HListElement that should be visible within the HListGroup .

setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
            HNavigable right)
```

Specified By:
setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) in interface HNavigable

See Also:
HNavigable

setGainFocusSound(HSound)

```
public void setGainFocusSound(HSound sound)
```

Specified By:
setGainFocusSound(HSound) in interface HNavigable

See Also:
HNavigable

setLoseFocusSound(HSound)

```
public void setLoseFocusSound(HSound sound)
```

Specified By:
setLoseFocusSound(HSound) in interface HNavigable

See Also:
HNavigable

setMove(int, HNavigable)

```
public void setMove(int keyCode, HNavigable target)
```

Specified By:
setMove(int, HNavigable) in interface HNavigable

See Also:
HNavigable

setMultiSelection(boolean)

```
public void setMultiSelection(boolean multiSelection)
```

Sets the HListGroup to allow multiple selection.

setNumVisibleElements(int)

```
public void setNumVisibleElements(int n)
```

Sets a hint as to the number of elements that will be visible within the HListGroup .

Parameters:
`n` - Number of visible elements.

org.havi.ui
HListGroupLayoutManager

Syntax

```
public class HListGroupLayoutManager implements java.awt.LayoutManager2

java.lang.Object
  |
  +--org.havi.ui.HListGroupLayoutManager
```

All Implemented Interfaces:

java.awt.LayoutManager, java.awt.LayoutManager2

Description

A HListGroupLayoutManager arranges HListElement in a vertical or horizontal list within the HListGroup container.

By default the HListGroupLayoutManager specifies that components should be laid out VERTICAL , and ignores the position of any added HListElement . The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
align	The orientation that the HListGroupLayoutManager lays out the HListElement within its associated HListGroup .	VERTICAL	setOrientation(int)	getOrientation()

Default parameter values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Fields

HORIZONTAL

```
public static final int HORIZONTAL
```

This value indicates that the list should be laid out horizontally

VERTICAL

```
public static final int VERTICAL
```

This value indicates that the list should be laid out vertically

Constructors

HListGroupLayoutManager()

```
public HListGroupLayoutManager()
```

Creates an HListGroupLayoutManager object. See the class description for details of constructor parameters and default values.

HListGroupLayoutManager(int)

```
public HListGroupLayoutManager(int align)
```

Creates an HListGroupLayoutManager object. See the class description for details of constructor parameters and default values.

Methods

addLayoutComponent(Component, Object)

```
public void addLayoutComponent(java.awt.Component comp, java.lang.Object constraints)
```

Note that HListGroupLayoutManager ignores the constraints parameter.

Specified By:
java.awt.LayoutManager2.addLayoutComponent(java.awt.Component, java.lang.Object) in interface java.awt.LayoutManager2

See Also:
`java.awt.LayoutManager2`

addLayoutComponent(String, Component)

```
public void addLayoutComponent(java.lang.String name, java.awt.Component comp)
```

Specified By:
java.awt.LayoutManager.addLayoutComponent(java.lang.String, java.awt.Component) in interface java.awt.LayoutManager

See Also:
`java.awt.LayoutManager`

getLayoutAlignmentX(Container)

```
public float getLayoutAlignmentX(java.awt.Container cont)
```

Specified By:
java.awt.LayoutManager2.getLayoutAlignmentX(java.awt.Container) in interface java.awt.LayoutManager2

See Also:
`java.awt.LayoutManager2`

getLayoutAlignmentY(Container)

```
public float getLayoutAlignmentY(java.awt.Container cont)
```

Specified By:
java.awt.LayoutManager2.getLayoutAlignmentY(java.awt.Container) in interface java.awt.LayoutManager2

See Also:
`java.awt.LayoutManager2`

getOrientation()

```
public int getOrientation()
```

Gets the current orientation of this HListGroupLayoutManager .

Returns:
the current orientation of this HListGroupLayoutManager .

invalidateLayout(Container)

```
public void invalidateLayout(java.awt.Container cont)
```

Specified By:
java.awt.LayoutManager2.invalidateLayout(java.awt.Container) in interface java.awt.LayoutManager2

See Also:
`java.awt.LayoutManager2`

layoutContainer(Container)

```
public void layoutContainer(java.awt.Container parent)
```

Specified By:
java.awt.LayoutManager.layoutContainer(java.awt.Container) in interface java.awt.LayoutManager

See Also:
`java.awt.LayoutManager`

maximumLayoutSize(Container)

```
public java.awt.Dimension maximumLayoutSize(java.awt.Container cont)
```

Specified By:
java.awt.LayoutManager2.maximumLayoutSize(java.awt.Container) in interface java.awt.LayoutManager2

See Also:
`java.awt.LayoutManager2`

minimumLayoutSize(Container)

```
public java.awt.Dimension minimumLayoutSize(java.awt.Container parent)
```

Specified By:
java.awt.LayoutManager.minimumLayoutSize(java.awt.Container) in interface java.awt.LayoutManager

See Also:
`java.awt.LayoutManager`

preferredLayoutSize(Container)

```
public java.awt.Dimension preferredLayoutSize(java.awt.Container parent)
```

Specified By:
java.awt.LayoutManager.preferredLayoutSize(java.awt.Container) in interface java.awt.LayoutManager

See Also:

java.awt.LayoutManager

removeLayoutComponent(Component)

```
public void removeLayoutComponent(java.awt.Component comp)
```

Specified By:
java.awt.LayoutManager.removeLayoutComponent(java.awt.Component) in interface java.awt.LayoutManager

See Also:
java.awt.LayoutManager

setOrientation(int)

```
public void setOrientation(int orientation)
```

Sets the orientation for this layout manager of this HListGroupLayoutManager .

Parameters:
orientation, - either HORIZONTAL or VERTICAL .

org.havi.ui
HLook

Syntax
```
public interface HLook extends java.lang.Cloneable
```

All Superinterfaces:
java.lang.Cloneable

All Known Implementing Classes:
HAnimateLook, HGraphicLook, HTextLook, HSinglelineEntryLook, HRangeLook

Description
The HLook interface defines the "look" of a component and may be regarded as a mechanism to allow a "pluggable" paint method to be attached to the component. That is instead of having to subclass the whole component to change its look, it is possible to simply implement an HLook that will render the component "look" and then associate this HLook implementation with the component.

The showLook(Graphics, HVisible, int) method of the HLook interface will be called by the havi.ui framework in response to the paint method of the HVisible being called by the AWT lightweight component framework. Applications should simply invoke the component repaint method as in normal AWT, rather than calling the showLook(Graphics, HVisible, int) method directly.

The conditions under which the showLook(Graphics, HVisible, int) method shall be invoked, include:

- If the class also implements HNavigable , then the showLook(Graphics, HVisible, int) method shall be invoked when the component receives either FocusEvent.FOCUSED_GAINED or FocusEvent.FOCUS_LOST events.
- If the class also implements HActionable or HSwitchable , then the showLook(Graphics, HVisible, int) method shall be invoked when the component receives an ActionEvent.ACTION_PERFORMED.
- If the class also implements HValue , then the showLook(Graphics, HVisible, int) method shall be invoked when the component receives an HValueChangeEvent .
- If the implementing class is an HVisible , then the showLook(Graphics, HVisible, int) method shall be invoked when content is set on that HVisible .

The HAVi UI provides a number of classes implementing the HLook interface. Applications wishing to provide their own HLook may directly implement this interface or may subclass those provided by the platform.

Implementations of HLook should use

- A default border horizontal spacing of 2 pixels.
- A default border vertical spacing of 2 pixels.
- The foreground color of its associated HVisible (using the java.awt.Component method getForeground) as a basis to determine the Color to render the border.

See Also:
setLook(HLook), paint(Graphics), HTextLook, HGraphicLook, HAnimateLook, HRangeLook, HSinglelineEntryLook, HMultilineEntryLook

Methods

getHorizontalBorderSpacing()

```
public int getHorizontalBorderSpacing()
```

Returns the horizontal spacing used for the border, as a number of pixels.

Returns:
The horizontal spacing of the border in pixels.

getMaximumSize(HVisible)

```
public java.awt.Dimension getMaximumSize(HVisible hvisible)
```

Gets the maximum size of this HLook . This size may be determined by retrieving associated content from the HVisible , calculating the size of this content and adding any additional dimensions that it may require for border decoration etc. If content is unavailable then the HLook should return the current size of the HVisible , as given by getSize().

Parameters:
`hvisible` - HVisible to which this HLook is attached.

Returns:
A dimension object indicating this HLook maximum size.

See Also:
getMaximumSize()

getMinimumSize(HVisible)

```
public java.awt.Dimension getMinimumSize(HVisible hvisible)
```

Gets the minimum size of this HLook . This size may be determined by retrieving associated content from the HVisible , calculating the size of this content and adding any additional dimensions that it may require for border decoration etc. If content is unavailable then the HLook should return the current size of the HVisible , as given by getSize().

Parameters:
`hvisible` - HVisible to which this HLook is attached.

Returns:
A dimension object indicating this HLook minimum size.

See Also:
getMinimumSize()

getPreferredSize(HVisible)

```
public java.awt.Dimension getPreferredSize(HVisible hvisible)
```

Gets the preferred size of this HLook . This size may be determined by retrieving associated content from the HVisible , calculating the size of this content and adding any additional dimensions that it may require for border decoration etc. If content is unavailable then the HLook should return the current size of the HVisible , as given by getSize().

Parameters:
`hvisible` - HVisible to which this HLook is attached.

Returns:
A dimension object indicating this HLook preferred size.

See Also:
getPreferredSize()

getVerticalBorderSpacing()

```
public int getVerticalBorderSpacing()
```

Returns the vertical spacing used for the border, as a number of pixels.

Returns:
The vertical spacing of the border in pixels.

setHorizontalBorderSpacing(int)

```
public void setHorizontalBorderSpacing(int space)
```

Sets the horizontal spacing the HLook should use for drawing any border decoration.

Parameters:
`space` - The horizontal spacing in pixels of the border, values less than zero shall be treated as zero. The default value is two pixels.

setVerticalBorderSpacing(int)

```
public void setVerticalBorderSpacing(int space)
```

Sets the vertical spacing the HLook should use for drawing any border decoration.

Parameters:
`space` - The vertical spacing in pixels of the border, values less than zero shall be treated as zero. The default value is two pixels.

showLook(Graphics, HVisible, int)

```
public void showLook(java.awt.Graphics g, HVisible visible, int state)
```

Abstract method showLook(Graphics, HVisible, int) to draw the look for a Component . The showLook(Graphics, HVisible, int) method is responsible for repainting the entire HVisible component, including its background, subject to the clipRect of the Graphics object passed to it. The showLook(Graphics, HVisible, int) method should not modify the clipRect of the Graphics object that is passed to it.

Any resources **explicitly** associated with an HLook should be loaded by the HLook during its creation, etc., or via its setXXX() methods. Note that the "standard" looks don't load content by default.

This method is called by the widget framework and should not be directly called by an application.

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.

org.havi.ui
HMatte

Syntax

```
public interface HMatte
```

All Known Implementing Classes:

HImageMatte, HFlatMatte, HFlatEffectMatte, HImageEffectMatte

Description

HMatte is the base interface for all matte classes.

Where pixels in a component already have an alpha value (e.g. from an image), the alpha value from the component and the alpha value from the HMatte are multiplied together to obtain the actual alpha value to be used for that pixel.

The final displayed value of the component and its HMatte is obviously subject to the capabilities of the underlying hardware platform.

org.havi.ui
HMatteException

Syntax

```
public class HMatteException extends HUIException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--HUIException
                    |
                    +--org.havi.ui.HMatteException
```

All Implemented Interfaces:

java.io.Serializable

Description

An HMatteException is an exception that is thrown when a Component is unable to support the desired HMatte effect. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

See Also:

HMatteLayer

Constructors

HMatteException()

```
public HMatteException()
```

org.havi.ui
HMatteLayer

Syntax
```
public interface HMatteLayer
```

All Known Implementing Classes:
HComponent, HContainer

Description
This HMatteLayer interface enables the presentation of components, together with an associated HMatte , for matte compositing.

See Also:
HMatte

Methods

getMatte()

```
public HMatte getMatte()
```

Returns the currently associated HMatte , from this component.

Returns:
the currently associated HMatte , from this component.

setMatte(HMatte)

```
public void setMatte(HMatte m)
```

Applies a HMatte to this component, for matte compositing.

Parameters:
`m` - The HMatte to be applied to this component -- note that only one matte may be associated with the component, thus any previous matte will be replaced. If m is null, then the component shall be treated as if it had a fully opaque HFlatMatte associated with it (value 1.0) -- this is the default value.

Throws:
HMatteException - if the HMatte cannot be associated with the component. For example, if the component is associated with an already running HFlatEffectMatte or HImageEffectMatte .

See Also:
HMatte

org.havi.ui
HMultilineEntry

Syntax

```
public class HMultilineEntry extends HSinglelineEntry

java.lang.Object
  |
  +--java.awt.Component
        |
        +--HComponent
              |
              +--HVisible
                    |
                    +--HSinglelineEntry
                          |
                          +--org.havi.ui.HMultilineEntry
```

All Implemented Interfaces:

HKeyboardInputPreferred, HMatteLayer, HNavigable, HState, HValue, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

The HMultilineEntry class is used to receive multiple lines of alphanumeric entry from the user. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	Width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
height	Height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Parameter	Description	Default value	Set method	Get method
text	The text within this HMultilineEntry , to be used as the displayed and editable content, for both the NORMAL_STATE and FOCUS_STATE states.	null	setTextContent(String, int)	getTextContent(int)
maxChars	The maximum number of characters allowed (per line) in this HMultilineEntry .	16 characters	setMaxCharsPerLine(int)	getMaxCharsPerLine()
maxLines	The maximum number of lines allowed in this HMultilineEntry .	4 lines	setMaxLines(int)	getMaxLines()
font	The font to be used for this component.	---	java.awt.Component#setFont.	java.awt.Component#getFont.
color	The color to be used for this component.	---	java.awt.Component#setForeground.	java.awt.Component#getForeground.

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The default "look" for this class.	A platform specific HMultilineEntryLook	setDefaultLook(HSinglelineEntryLook)	getDefaultLook()
The "look" for this object.	The HMultilineEntryLook returned from HMultilineEntry.getDefaultLook when this object was created.	setLook(HLook)	getLook()
The gain focus sound.	null	setGainFocusSound(HSound)	getGainFocusSound()
The lose focus sound.	null	setLoseFocusSound(HSound)	getLoseFocusSound()
Caret position	At the end of the current text string	setCaretCharPosition(int)	getCaretCharPosition()
Input type	INPUT_ANY	setType(int)	getType()
Password protection (the echo character)	Entry is "clear", i.e. not password protected.	setEchoChar(char)	getEchoChar() / echoCharIsSet()

Constructors

HMultilineEntry()

```
public HMultilineEntry()
```

Creates an HMultilineEntry object. See the class description for details of constructor parameters and default values.

HMultilineEntry(int, int)

```
public HMultilineEntry(int maxChars, int maxLines)
```

Creates an HMultilineEntry object. See the class description for details of constructor parameters and default values.

HMultilineEntry(int, int, int, int, int, int)

```
public HMultilineEntry(int x, int y, int width, int height, int maxChars, int maxLines)
```

Creates an HMultilineEntry object. See the class description for details of constructor parameters and default values.

HMultilineEntry(String, int, int, Font, Color)

```
public HMultilineEntry(java.lang.String text, int maxChars, int maxLines, java.awt.
            Font font, java.awt.Color color)
```

Creates an HMultilineEntry object. See the class description for details of constructor parameters and default values.

HMultilineEntry(String, int, int, int, int, int, int, Font, Color)

```
public HMultilineEntry(java.lang.String text, int x, int y, int width, int height,
            int maxChars, int maxLines, java.awt.Font font, java.awt.Color color)
```

Creates an HMultilineEntry object. See the class description for details of constructor parameters and default values.

Methods

caretNextLine()

```
public void caretNextLine()
```

Move the caret to the same column position on the previous line. If the caret would be past the end of the text on the line the new caret position will be at the end of the line.

caretPreviousLine()

```
public void caretPreviousLine()
```

Move the caret to the same column position on the next line. If the caret would be past the end of the text on the line the new caret position will be at the end of the line.

getCaretLinePosition()

```
public int getCaretLinePosition()
```

Gets the current line position for the text insertion point.

Returns:
the line position of the text insertion caret.

getDefaultLook()

```
public static HSinglelineEntryLook getDefaultLook()
```

Returns the currently set default look for HMultilineEntry components.

Returns:
The HMultilineEntryLook (as an HSinglelineEntryLook) that is used by default when creating a new HMultilineEntry component.

getMaxLines()

```
public int getMaxLines()
```

Get maximum number of lines of text in this widget.

setCaretCharPosition(int)

```
public void setCaretCharPosition(int position)
```

Sets the column position of the text insertion caret for this text component, within the current line.

Overrides:
setCaretCharPosition(int) in class HSinglelineEntry

Parameters:
`position` - the new column position of the text insertion caret in the current line.

setCaretLinePosition(int)

```
public void setCaretLinePosition(int position)
```

Sets the line position of the text insertion caret for this text component, in the current column.

Parameters:
`position` - the new line position of the text insertion caret, in the current column.

setDefaultLook(HSinglelineEntryLook)

```
public static void setDefaultLook(HSinglelineEntryLook look)
```

Sets the default look for all HMultilineEntry Components.

Parameters:
`look` - The look that will be used by default when creating a new HMultilineEntry component.

Throws:
HInvalidLookException - If the HLook is not an HMultilineEntryLook .

setLook(HLook)

```
public void setLook(HLook hlook)
```

Sets the HLook for this component.

Overrides:
setLook(HLook) in class HSinglelineEntry

Parameters:
`hlook` - The HLook that is to be used for this component.

Throws:
HInvalidLookException - If the Look is not an HMultilineEntryLook .

setMaxLines(int)

```
public void setMaxLines(int maxLines)
```

Set maximum number of lines of text in this widget.

Returns:

maxLines the maximum number of lines of text in this widget

org.havi.ui
HMultilineEntryLook

Syntax
```
public class HMultilineEntryLook extends HSinglelineEntryLook

java.lang.Object
  |
  +--HSinglelineEntryLook
        |
        +--org.havi.ui.HMultilineEntryLook
```

All Implemented Interfaces:
java.lang.Cloneable, HLook

Description
The HMultilineEntryLook class is used by the HMultilineEntry component to display the entering of text. This look will be provided by the platform and the exact way in which it is rendered will be platform dependant. However, the HMultilineEntryLook associated with a given HVisible should use:

- java.awt.Component#getForeground() to determine the Color to render the rectangle.
- Border widths (as set by setHorizontalBorderSpacing / setVerticalBorderSpacing) of 2 pixels by default.

This is the default look that is used by HMultilineEntry and its subclasses. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

See Also:
HMultilineEntry

Constructors

HMultilineEntryLook()
```
public HMultilineEntryLook()
```
Creates an HMultilineEntryLook that will be used for entering text.

Methods

getHorizontalBorderSpacing()

```
public int getHorizontalBorderSpacing()
```

Overrides:
getHorizontalBorderSpacing() in class HSinglelineEntryLook

See Also:
HLook

getMaximumSize(HVisible)

```
public java.awt.Dimension getMaximumSize(HVisible visible)
```

Overrides:
getMaximumSize(HVisible) in class HSinglelineEntryLook

See Also:
HLook

getMinimumSize(HVisible)

```
public java.awt.Dimension getMinimumSize(HVisible visible)
```

Overrides:
getMinimumSize(HVisible) in class HSinglelineEntryLook

See Also:
HLook

getPreferredSize(HVisible)

```
public java.awt.Dimension getPreferredSize(HVisible visible)
```

Overrides:
getPreferredSize(HVisible) in class HSinglelineEntryLook

See Also:
HLook

getVerticalBorderSpacing()

```
public int getVerticalBorderSpacing()
```

Overrides:
getVerticalBorderSpacing() in class HSinglelineEntryLook

See Also:
HLook

setHorizontalBorderSpacing(int)

```
public void setHorizontalBorderSpacing(int width)
```

Overrides:
setHorizontalBorderSpacing(int) in class HSinglelineEntryLook

See Also:
HLook

setVerticalBorderSpacing(int)

```
public void setVerticalBorderSpacing(int width)
```

Overrides:
setVerticalBorderSpacing(int) in class HSinglelineEntryLook

See Also:
HLook

showLook(Graphics, HVisible, int)

```
public void showLook(java.awt.Graphics g, HVisible visible, int state)
```

Draws the text that has been entered into the HMultilineEntry on screen.

Overrides:
showLook(Graphics, HVisible, int) in class HSinglelineEntryLook

See Also:
HLook

org.havi.ui
HNavigable

Syntax
```
public interface HNavigable
```

All Known Subinterfaces:
HActionable, HSwitchable, HValue

All Known Implementing Classes:
HIcon, HRange, HText, HAnimation

Description
The HNavigable interface is implemented by HAVi UI components that can be navigated to by the user (i.e. they can gain focus). The mechanism of producing focus events is system specific.

Hence, a component implementing the HNavigable interface has two states - NORMAL_STATE (unfocused) and FOCUS_STATE .

The state transitions for an unfocused HNavigable in the NORMAL_STATE are as follows:

1. The HNavigable receives focus when it receives a FOCUS_GAINED java.awt.event.focusEvent.
2. The HNavigable modifies its interaction state to FOCUS_STATE .
3. The HNavigable repaints itself to show any visual change in appearance, due to the associated HLook .
4. Any sound associated with the HNavigable by the setGainFocusSound(HSound) method shall be played. Note that it is not guaranteed that this sound will be played to completion, eg due to further user-interaction, etc.
5. Any FocusListeners are then called.

The state transitions for a focused HNavigable in the FOCUS_STATE are as follows:

1. The HNavigable looses focus when it receives a FOCUS_LOST java.awt.event.focusEvent.
2. The HNavigable modifies its interaction state to NORMAL_STATE .
3. The HNavigable repaints itself to show any visual change in appearance, due to the associated HLook .
4. Any sound associated with the HNavigable by the setLoseFocusSound(HSound) method shall be played. Note that it is not guaranteed that this sound will be played to completion, eg due to further user-interaction, etc.
5. Any FocusListeners are then called.

An HNavigable also has an arbitrary focus traversal table associated with it (see setMove(int, HNavigable) and getMove(int)). This mechanism allows the four-way focus behaviour of a set of components to be set (see setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) , setMove(int, HNavigable) and getMove(int)).

Subclasses of java.awt.Component which implement HNavigable should by default enable java.awt.event.FocusEvents. Whilst subclasses of java.awt.Component implementing HNavigable may enable additional java.awt.AWTEvents, applications should assume that such classes only generate java.awt.event.FocusEvents and should use the standard AWT mechanisms to enable additional events to be generated, if required.

In particular, the following classes implementing HNavigable should all generate java.awt.event.FocusEvent's

- HAnimation
- HIcon
- HText
- HRange
- HGraphicButton
- HTextButton
- HToggleButton
- HListElement
- HSinglelineEntry
- HRangeValue

It is a valid implementation option for subclasses of java.awt.Component implementing HNavigable to override the protected processEvent and/or processFocusEvent methods.

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

Methods

getGainFocusSound()

```
public HSound getGainFocusSound()
```

Get the sound associated with the gaining focus event.

Returns:
The sound played when the component gains focus. If no sound is associated with gaining focus, then null shall be returned.

getLoseFocusSound()

```
public HSound getLoseFocusSound()
```

Get the sound associated with the lost focus event.

Returns:
The sound played when the component looses focus. If no sound is associated with loosing focus, then null shall be returned.

getMove(int)

```
public HNavigable getMove(int keyCode)
```

Provides the HNavigable object that is navigated to when a particular key is pressed.

Parameters:
`keyCode` - The key code of the pressed key.

Returns:
Returns the HNavigable object, or if no HNavigable is associated with the keyCode then returns null.

isSelected()

```
public boolean isSelected()
```

Indicates if this component has focus.

Returns:
TRUE if the component has focus, otherwise returns FALSE.

requestFocus()

```
public void requestFocus()
```

The requestFocus method defers to the java.awt.Component requestFocus method in those situations that the class is a subclass of java.awt.Component and also implements the HNavigable interface.

The behaviour of this method when the HNavigable interface is implemented by classes which are not subclasses of java.awt.Component should be specified within those classes.

See Also:

`java.awt.Component.requestFocus()`

setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
            HNavigable right)
```

Set the focus control for a HNavigable component. Note setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) is a convenience function for application programmers where a standard up, down, left and right focus traversal between widgets is required.

Note setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) is equivalent to multiple calls to setMove(int, HNavigable) , where the key codes VK_UP, VK_DOWN, VK_LEFT, VK_RIGHT are used.

Note that this API does not prevent the creation of "isolated" HNavigable components --- authors should endeavour to avoid confusing the user.

Parameters:

`up` - The HNavigable component to move to, when the user performs a VK_UP. If there is no HNavigable component to move "up" to, then null should be specified.

`down` - The HNavigable component to move to, when the user performs a VK_DOWN. If there is no HNavigable component to move "down" to, then null should be specified.

`left` - The HNavigable component to move to, when the user performs a VK_LEFT. If there is no HNavigable component to move "left" to, then null should be specified.

`right` - The HNavigable component to move to, when the user performs a VK_RIGHT. If there is no HNavigable component to move "right" to, then null should be specified.

setGainFocusSound(HSound)

```
public void setGainFocusSound(HSound sound)
```

Associate a sound with gaining focus, ie when the HNavigable receives a java.awt.event.FocusEvent event of type FOCUS_GAINED. This sound will start to be played when an object implementing this interface gains focus. It is not guaranteed to be played to completion. If the object implementing this interface looses focus before the audio completes playing, the audio will be truncated. Applications wishing to ensure the audio is always played to completion must implement special logic to slow down the focus transitions.

By default, an HNavigable object does not have any gain focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

Parameters:

`sound` - the sound to be played, when the component looses focus. If sound content is already set, the original content is replaced. To remove the sound specify a null HSound .

setLoseFocusSound(HSound)

```
public void setLoseFocusSound(HSound sound)
```

Associate a sound with losing focus, ie when the HNavigable receives a java.awt.event.FocusEvent event of type FOCUS_LOST. This sound will start to be played when an object implementing this interface looses focus. It is not guaranteed to be played to completion. It is implementation dependent whether and when this sound will be truncated by any gainFocusSound for the next object to gain focus.

By default, an HNavigable object does not have any loose focus sound associated with it.

Note that the ordering of playing sounds is dependent on the order of the focus lost, gained events.

Parameters:

`sound` - the sound to be played, when the component looses focus. If sound content is already set, the original content is replaced. To remove the sound specify a null HSound .

setMove(int, HNavigable)

```
public void setMove(int keyCode, HNavigable target)
```

Defines the navigation path from the current HNavigable to another HNavigable when a particular key is pressed.

Note that setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) is equivalent to multiple calls to setMove(int, HNavigable) , where the key codes VK_UP, VK_DOWN, VK_LEFT, VK_RIGHT are used.

Parameters:

`keyCode` - The key code of the pressed key.

`target` - The target HNavigable object that should be navigated to. If a target is to be removed from a particular navigation path, then null should be specified.

org.havi.ui
HNoInputPreferred

Syntax
```
public interface HNoInputPreferred
```

All Known Implementing Classes:
HStaticIcon, HStaticRange, HStaticText, HStaticAnimation

Description
A component which implements HNoInputPreferred indicates that the user cannot navigate to this component. In some cases a component which implements this interface will be extended, and that component will implement another "XxxInputPreferred" interface. In all cases, these other interfaces take precedence.

org.havi.ui
HPermissionDeniedException

Syntax

```
public class HPermissionDeniedException extends java.lang.Exception

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.havi.ui.HPermissionDeniedException
```

All Implemented Interfaces:

java.io.Serializable

Description

Thrown when an application calls a method which it does not have permission to do at that time. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter value parameter values exposed in the constructors:

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Constructors

HPermissionDeniedException()

```
public HPermissionDeniedException()
```

Default constructor for the exception

HPermissionDeniedException(String)

```
public HPermissionDeniedException(java.lang.String reason)
```

Constructor for the exception with a specified reason

Parameters:

`reason` - the reason why the exception was raised

org.havi.ui
HRange

Syntax

```
public class HRange extends HStaticRange implements HNavigable
```

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--HComponent
              |
              +--HVisible
                    |
                    +--HStaticRange
                          |
                          +--org.havi.ui.HRange
```

Direct Known Subclasses:

HRangeValue

All Implemented Interfaces:

HMatteLayer, HNavigable, HNoInputPreferred, HState, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

The HRange component is used for displaying a value which is within a fixed range. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
wth	Width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Parameter	Description	Default value	Set method	Get method
height	Height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
orientation	The "orientation" of the range object.	OR_HORIZ	setOrientation(int)	getOrientation()
minimum	The minimum value that can be returned by this range object.	0	setRange(int, int)	getMinValue()
maximum	The maximum value that can be returned by this range object.	100	setRange(int, int)	getMaxValue()
value	The current value returned by this range object.	0	setValue(int)	getValue()

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The default "look" for this class.	A platform specific HRangeLook	setDefaultLook(HRangeLook)	getDefaultLook()
The "look" for this object.	The HGraphicLook returned from HRange.getDefaultLook when this object was created.	setLook(HLook)	getLook()
The gain focus sound.	null	setGainFocusSound(HSound)	getGainFocusSound()
The lose focus sound.	null	setLoseFocusSound(HSound)	getLoseFocusSound()

Constructors

HRange()

```
public HRange()
```

Creates an HRange object. See the class description for details of constructor parameters and default values.

HRange(int, int, int, int)

```
public HRange(int orientation, int minimum, int maximum, int value)
```

Creates an HRange object. See the class description for details of constructor parameters and default values.

HRange(int, int, int, int, int, int, int, int)

```
public HRange(int orientation, int minimum, int maximum, int value, int x, int y,
              int width, int height)
```

Creates an HRange object. See the class description for details of constructor parameters and default values.

Methods

getDefaultLook()

```
public static HRangeLook getDefaultLook()
```

Returns the currently set default HLook for HRange components.

Returns:
The HLook that is used by default when creating a new HRange component.

getGainFocusSound()

```
public HSound getGainFocusSound()
```

Specified By:
getGainFocusSound() in interface HNavigable

See Also:
HNavigable

getLoseFocusSound()

```
public HSound getLoseFocusSound()
```

Specified By:
getLoseFocusSound() in interface HNavigable

See Also:
HNavigable

getMove(int)

```
public HNavigable getMove(int keyCode)
```

Specified By:
getMove(int) in interface HNavigable

See Also:
HNavigable

isFocusTraversable()

```
public boolean isFocusTraversable()
```

By default classes implementing HNavigable are focus-traversable.

Overrides:
isFocusTraversable() in class HVisible

Returns:
true

See Also:
`java.awt.Component.isFocusTraversable()`

isSelected()

```
public boolean isSelected()
```

Specified By:
isSelected() in interface HNavigable

See Also:
HNavigable

processEvent(AWTEvent)

```
protected void processEvent(java.awt.AWTEvent evt)
```

It is a valid implementation option for HRange to override the protected processEvent method.

Overrides:
java.awt.Component.processEvent(java.awt.AWTEvent) in class java.awt.Component

See Also:
`java.awt.Component.processEvent(AWTEvent)`

processFocusEvent(FocusEvent)

```
protected void processFocusEvent(java.awt.event.FocusEvent evt)
```

It is a valid implementation option for HRange to override the protected processFocusEvent method.

Overrides:
java.awt.Component.processFocusEvent(java.awt.event.FocusEvent) in class java.awt.Component

See Also:
`java.awt.Component.processFocusEvent(FocusEvent)`

requestFocus()

```
public void requestFocus()
```

Specified By:
requestFocus() in interface HNavigable

Overrides:
java.awt.Component.requestFocus() in class java.awt.Component

See Also:
`java.awt.Component.requestFocus()`

setDefaultLook(HRangeLook)

```
public static void setDefaultLook(HRangeLook look)
```

Sets the default HLook for all HRange Components.

Parameters:
`look` - The HLook that will be used by default when creating a new HRange component.

Throws:
HInvalidLookException - If the HLook is not an HRangeLook .

setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
            HNavigable right)
```

Specified By:
setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) in interface HNavigable

See Also:
HNavigable

setGainFocusSound(HSound)

```
public void setGainFocusSound(HSound sound)
```

Specified By:
setGainFocusSound(HSound) in interface HNavigable

See Also:
HNavigable

setLoseFocusSound(HSound)

```
public void setLoseFocusSound(HSound sound)
```

Specified By:
setLoseFocusSound(HSound) in interface HNavigable

See Also:
HNavigable

setMove(int, HNavigable)

```
public void setMove(int keyCode, HNavigable target)
```

Specified By:
setMove(int, HNavigable) in interface HNavigable

See Also:
HNavigable

org.havi.ui
HRangeLook

Syntax
```
public class HRangeLook implements HLook

java.lang.Object
  |
  +--org.havi.ui.HRangeLook
```

All Implemented Interfaces:
java.lang.Cloneable, HLook

Description
The HRangeLook class displays a slider type range control on screen. This look will be provided by the platform and the exact way in which it is rendered will be platform dependant. However, the HRangeLook associated with a given HVisible should use:

- java.awt.Component#getForeground() to determine the Color to render the rectangle.
- Border widths (as set by setHorizontalBorderSpacing / setVerticalBorderSpacing) of 2 pixels by default.

This is the default look that is used by HStaticRange and its subclasses. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

See Also:
HRange

Constructors

HRangeLook()
```
public HRangeLook()
```
Creates an HRangeLook that will be used for rendering a range of values on screen.

Methods

getHorizontalBorderSpacing()

```
public int getHorizontalBorderSpacing()
```

Specified By:
getHorizontalBorderSpacing() in interface HLook

See Also:
HLook

getMaximumSize(HVisible)

```
public java.awt.Dimension getMaximumSize(HVisible visible)
```

Specified By:
getMaximumSize(HVisible) in interface HLook

See Also:
HLook

getMinimumSize(HVisible)

```
public java.awt.Dimension getMinimumSize(HVisible visible)
```

Specified By:
getMinimumSize(HVisible) in interface HLook

See Also:
HLook

getPreferredSize(HVisible)

```
public java.awt.Dimension getPreferredSize(HVisible visible)
```

Specified By:
getPreferredSize(HVisible) in interface HLook

See Also:
HLook

getVerticalBorderSpacing()

```
public int getVerticalBorderSpacing()
```

Specified By:
getVerticalBorderSpacing() in interface HLook

See Also:
HLook

setHorizontalBorderSpacing(int)

```
public void setHorizontalBorderSpacing(int width)
```

Specified By:
setHorizontalBorderSpacing(int) in interface HLook

See Also:
HLook

setVerticalBorderSpacing(int)

```
public void setVerticalBorderSpacing(int width)
```

Specified By:
setVerticalBorderSpacing(int) in interface HLook

See Also:
HLook

showLook(Graphics, HVisible, int)

```
public void showLook(java.awt.Graphics g, HVisible visible, int state)
```

Specified By:
showLook(Graphics, HVisible, int) in interface HLook

See Also:
HLook

org.havi.ui
HRangeValue

Syntax

```
public class HRangeValue extends HRange implements HValue, HAdjustmentInputPreferred
```

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--HComponent
              |
              +--HVisible
                    |
                    +--HStaticRange
                          |
                          +--HRange
                                |
                                +--org.havi.ui.HRangeValue
```

All Implemented Interfaces:

HAdjustmentInputPreferred, HMatteLayer, HNavigable, HNoInputPreferred, HState, HValue, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

The HRangeValue class provides a slider, or range control. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	Width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
height	Height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Parameter	Description	Default value	Set method	Get method
orientation	The "orientation" of the range object.	OR_HORIZ	setOrientation(int)	getOrientation()
minimum	The minimum value that can be returned by this range object.	0	setRange(int, int)	getMinValue()
maximum	The maximum value that can be returned by this range object.	100	setRange(int, int)	getMaxValue()
value	The current value returned by this range object.	0	setValue(int)	getValue()

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The default "look" for this class.	A platform specific HRangeLook	setDefaultLook(HRangeLook)	getDefaultLook()
The "look" for this object.	The HGraphicLook returned from HRangeValue.getDefaultLook when this object was created.	setLook(HLook)	getLook()
The gain focus sound.	null	setGainFocusSound(HSound)	getGainFocusSound()
The lose focus sound.	null	setLoseFocusSound(HSound)	getLoseFocusSound()
The block increment for this object.	1 unit	setBlockIncrement(int)	getBlockIncrement()

Constructors

HRangeValue()

```
public HRangeValue()
```

Creates an HRangeValue object. See the class description for details of constructor parameters and default values.

HRangeValue(int, int, int, int)

```
public HRangeValue(int orientation, int minimum, int maximum, int value)
```

Creates an HRangeValue object. See the class description for details of constructor parameters and default values.

HRangeValue(int, int, int, int, int, int, int, int)

```
public HRangeValue(int orientation, int minimum, int maximum, int value, int x, int y,
            int width, int height)
```

Creates an HRangeValue object. See the class description for details of constructor parameters and default values.

Methods

addChangeListener(HValueChangeListener)

```
public void addChangeListener(HValueChangeListener l)
```

Specified By:
addChangeListener(HValueChangeListener) in interface HValue

See Also:
HValue

getBlockIncrement()

```
public int getBlockIncrement()
```

Get the block increment for this HRangeValue .

Returns:
the block increment value for this HRangeValue .

getChangeSound()

```
public HSound getChangeSound()
```

Specified By:
getChangeSound() in interface HValue

See Also:
HValue

getDefaultLook()

```
public static HRangeLook getDefaultLook()
```

Returns the currently set default look for HRangeValue components.

Returns:
The look that is used by default when creating a new HRangeValue component.

getGainFocusSound()

```
public HSound getGainFocusSound()
```

Specified By:
getGainFocusSound() in interface HNavigable

Overrides:
getGainFocusSound() in class HRange

See Also:
HNavigable

getLoseFocusSound()

```
public HSound getLoseFocusSound()
```

Specified By:
getLoseFocusSound() in interface HNavigable

Overrides:
getLoseFocusSound() in class HRange

See Also:
HNavigable

getMove(int)

```
public HNavigable getMove(int keyCode)
```

Specified By:
getMove(int) in interface HNavigable

Overrides:
getMove(int) in class HRange

See Also:
HNavigable

isFocusTraversable()

```
public boolean isFocusTraversable()
```

By default classes implementing HNavigable are focus-traversable.

Overrides:
isFocusTraversable() in class HRange

Returns:
true

See Also:
`java.awt.Component.isFocusTraversable()`

isSelected()

```
public boolean isSelected()
```

Specified By:
isSelected() in interface HNavigable

Overrides:
isSelected() in class HRange

See Also:
HNavigable

processChangeEvent(HValueChangeEvent)

```
protected void processChangeEvent(HValueChangeEvent evt)
```

Processes HValueChangeEvent enabled for this object. Subclasses of classes implementing processChangeEvent should call super.processChangeEvent to ensure that the change event is handled appropriately.

Parameters:
`evt` - the event to be processed.

See Also:
HValue

processEvent(AWTEvent)

```
protected void processEvent(java.awt.AWTEvent evt)
```

It is a valid implementation option for HRangeValue to override the protected processEvent method.

Overrides:
processEvent(AWTEvent) in class HRange

See Also:
`java.awt.Component.processEvent(AWTEvent)`

processFocusEvent(FocusEvent)

```
protected void processFocusEvent(java.awt.event.FocusEvent evt)
```

It is a valid implementation option for HRangeValue to override the protected processFocusEvent method.

Overrides:
processFocusEvent(FocusEvent) in class HRange

See Also:
`java.awt.Component.processFocusEvent(FocusEvent)`

processKeyEvent(KeyEvent)

```
protected void processKeyEvent(java.awt.event.KeyEvent evt)
```

It is a valid implementation option for HRangeValue to override the protected processKeyEvent method.

Overrides:
java.awt.Component.processKeyEvent(java.awt.event.KeyEvent) in class java.awt.Component

See Also:
`java.awt.Component.processKeyEvent(KeyEvent)`

removeChangeListener(HValueChangeListener)

```
public void removeChangeListener(HValueChangeListener l)
```

Specified By:
removeChangeListener(HValueChangeListener) in interface HValue

See Also:
HValue

setBlockIncrement(int)

```
public void setBlockIncrement(int increment)
```

Set the block increment for this HRangeValue .

Parameters:
`increment` - the value by which successive values of the HRangeValue should differ, for the minimum user variation. Values of increment less than one, shall be treated as one.

setChangeSound(HSound)

```
public void setChangeSound(HSound sound)
```

Specified By:
setChangeSound(HSound) in interface HValue

See Also:
HValue

setDefaultLook(HRangeLook)

```
public static void setDefaultLook(HRangeLook look)
```

Sets the default HLook for all HRangeValue Components.

Parameters:
`look` - The HLook that will be used by default when creating a new HRangeValue component.

Throws:
HInvalidLookException - If the HLook is not an HRangeLook .

setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
            HNavigable right)
```

Specified By:
setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) in interface HNavigable

Overrides:
setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) in class HRange

See Also:
HNavigable

setGainFocusSound(HSound)

```
public void setGainFocusSound(HSound sound)
```

Specified By:
setGainFocusSound(HSound) in interface HNavigable

Overrides:
setGainFocusSound(HSound) in class HRange

See Also:
HNavigable

setLoseFocusSound(HSound)

```
public void setLoseFocusSound(HSound sound)
```

Specified By:
setLoseFocusSound(HSound) in interface HNavigable

Overrides:
setLoseFocusSound(HSound) in class HRange

See Also:
HNavigable

setMove(int, HNavigable)

```
public void setMove(int keyCode, HNavigable target)
```

Specified By:
setMove(int, HNavigable) in interface HNavigable

Overrides:
setMove(int, HNavigable) in class HRange

See Also:
HNavigable

org.havi.ui
HScene

Syntax

```
public class HScene extends HContainer

java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--HContainer
                    |
                    +--org.havi.ui.HScene
```

All Implemented Interfaces:

HMatteLayer, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

An HScene is a container representing the displayable area on-screen within which the application can display itself and thus interact with the user. However, HScene does not paint itself on-screen, only its added "child" components and hence there is no requirement to allocate "pixels" to the HScene directly --- its only immediate graphical effect is to "clip" its child components. Hence, HScene may be regarded as a simple connection to the window management policy within the device, acting as a "screen resource reservation mechanism" denoting the area within which an application may wish to present a component, at some point in the future.

For all interoperable applications, the HScene is considered the top-level component of the application. No parent component to an HScene should be accessible to applications. Interoperable applications should not use the getParent method in HScene , since results are implementation dependent and valid implementations may generate a run-time error.

In terms of delegation, the HScene shall behave like a Window with a native peer implementation, in that it will not appear to delegate any functionality to any parent object. Components which do not specify default characteristics inherit default values transitively from their parent objects. Therefore, the implementation of HScene must have valid defaults defined for all characteristics, e.g. Font, foreground Color, background Color, ColorModel, Cursor and Locale.

The HScene is a consumer of KeyEvents, e.g. for application-wide "shortcut-keys". HScene implementations should by default enable both java.awt.event.FocusEvents and java.awt.event.KeyEvents. Whilst implementations of HScene may enable additional java.awt.AWTEvents, applications should assume only java.awt.event.FocusEvents and java.awt.event.KeyEvents are generated, and should use the standard AWT mechanisms to enable additional events to be generated, if required. It is a valid option for implementations of HScene to override the protected methods: processEvent, processFocusEvent and processKeyEvent.

Applications may use the standard AWT mechanisms to enable additional events, but should not disable either java.awt.event.FocusEvents or java.awt.event.KeyEvents on the HScene .

It is the responsibility of the application designer to ensure that the relevant KeyEvents used for "shortcut keys" are not consumed by any child Component. Implementations of the standard HAVi UI widgets which process java.awt.event.KeyEvent's shall not consume any KeyEvent, thus allowing their use as a shorcut key.

The mechanism by which input events are passed to the HScene and its component hierarchy is not specified.

There is no public constructor for HScene , it is constructed by a HSceneFactory . Only one HScene per HGraphicsDevice can be acquired at any one time for each application.

When the entire application looses the user's focus, then this is indicated by the HScene object loosing focus, e.g. if the application is iconised. When the entire application regains the user's focus, then this is indicated by the HScene object regaining focus, e.g. if the application is de-iconised. If the entire application has its location, or extent modified, then these events should be signalled by the appropriate mechanisms being applied to the HScene object.

Finally, if the HScene object requestFocus method is invoked, then this should be treated as a request to make the entire application visible and ready for user input, e.g. by expanding an icon, or changing the stacking order between competing overlapping applications. The decision as to whether the HScene (application) gains the user's focus is entirely platform specific in terms of policy, etc. as to whether, or whenever the HScene might gain the user's focus. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Description	Default value	Set method	Get method
Visibility of the HScene	false	setVisible	isVisible
Activity of associated short-cuts	Shortcuts are active	enableShortcuts(boolean)	---

Constructors

HScene()

```
protected HScene()
```

It is not intended that applications should directly construct HScene objects. HScene objects should be constructed via the HSceneFactory classes factory methods. Creates an HScene object. See the class description for details of constructor parameters and default values.

See Also:
HSceneFactory

Methods

addShortcut(int, HActionable)

```
public void addShortcut(int keyCode, HActionable comp)
```

Generating the defined java.awt.KeyEvent or HRcEvent keycode causes the specified widget to become actioned -- potentially any keyCode may have a shortcut associated with it. The short-cut will only be made available if the HActionable widget is contained within the HScene .

Note that a maximum of one HActionable may be associated with a given keyCode. Calling addShortcut(int, HActionable) with the same HActionable will result in the previous short cut being removed. A short cut can be set on an invisible HActionable and therefore it is possible to provide short-cuts that have no user representation.

If the relevant keyCode is received by the HScene , then the HActionable will be actioned by the HScene sending it a VK_ACTION .

Parameters:

`keyCode` - The keycode that represents the short cut. If keycode is java.awt.event. KeyEvent#VK_UNDEFINED, then the shortcut will not be added.

`comp` - The actionable component that will be actioned.

addWindowListener(WindowListener)

```
public void addWindowListener(java.awt.event.WindowListener wl)
```

Add a listener to receive any java.awt.event.WindowEvents

Parameters:

`wl` - The java.awt.event.WindowListener to be notified of any java.awt.event.WindowEvents.

enableShortcuts(boolean)

```
public void enableShortcuts(boolean enable)
```

Enables or disables all short cuts that are currently set on the Scene. To enable or disable a single shortcut use addShortcut(int, HActionable) or removeShortcut(int) .

Note enableShortcuts(boolean) does not remove existing added HScene shortcuts - they are merely disabled and may be subsequently re-enabled with enableShortcuts(boolean) .

Parameters:

`enable` - a value of true indicates all shortcuts are to be enabled, and a value of false indicates all shortcuts are to be disabled.

getAllShortcutKeycodes()

```
public int[] getAllShortcutKeycodes()
```

Returns all keycodes added in the HScene as shortcuts.

Returns:

all keycodes added in the HScene as shortcuts, there are no ordering guarantees.

getPixelCoordinatesHScreenRectangle(Rectangle)

```
public HScreenRectangle getPixelCoordinatesHScreenRectangle(java.awt.Rectangle r)
```

Returns an HScreenRectangle which corresponds to the graphics (AWT) pixel area specified by the parameter in this HScene (i.e. within the HScene's coordinate space).

Parameters:

`r` - the AWT pixel area within this HScene (i.e. within the HScene's coordinate space), specified as an java.awt.Rectangle.

Returns:

an HScreenRectangle which corresponds to the graphics (AWT) pixel area specified by the parameter in this HScene (i.e. within the HScene's coordinate space).

getSceneTemplate()

```
public HSceneTemplate getSceneTemplate()
```

Return a HSceneTemplate describing this HScene . This template can be queried in order to obtain the size & position of the HScene in screen coordinates and the display device used for the HScene , etc.

Returns:
a HSceneTemplate describing of the HScene .

getShortcutKeycode(HActionable)

```
public int getShortcutKeycode(HActionable comp)
```

Returns the keycode associated with the specified HActionable component.

Parameters:
`comp` - the HActionable to return the keycode that it is associated with.

Returns:
the keycode associated with the specified HActionable component, if it is currently a valid shortcut "target", otherwise return java.awt.event.KeyEvent#VK_UNDEFINED.

isEnableShortcuts()

```
public boolean isEnableShortcuts()
```

Returns the status of all short cuts that are currently set on the HScene.

Returns:
true if shortcuts are enabled, false otherwise.

See Also:
enableShortcuts(boolean)

isVisible()

```
public boolean isVisible()
```

Determines if the HScene (or more properly its added child components) is Visible. Initially an HScene is invisible.

Overrides:
java.awt.Component.isVisible() in class java.awt.Component

Returns:
true if the HScene is visible; false otherwise.

paint(Graphics)

```
public void paint(java.awt.Graphics g)
```

HScene objects override the paint method (define'd in java.awt.Component) since HScene do not paint themselves on-screen, simply their added "child" components.

Overrides:
java.awt.Container.paint(java.awt.Graphics) in class java.awt.Container

processWindowEvent(WindowEvent)

```
protected void processWindowEvent(java.awt.event.WindowEvent we)
```

Process a java.awt.event.WindowEvent for this HScene .

Parameters:
`we` - the java.awt.event.WindowEvent to be processed.

removeShortcut(int)

```
public void removeShortcut(int keyCode)
```

Removes the specified short-cut key. if the specified short-cut key is not registered, the method has no effect

Parameters:

`keyCode` - The keycode that represents the short cut

removeWindowListener(WindowListener)

```
public void removeWindowListener(java.awt.event.WindowListener wl)
```

Remove a listener so that it no longer receives any java.awt.event.WindowEvents. if the specified listener is not registered, the method has no effect.

Parameters:

`wl` - The java.awt.event.WindowListener to be removed from notification of any java.awt.event. WindowEvents.

setVisible(boolean)

```
public void setVisible(boolean show)
```

Shows or hides this HScene (or more properly its added child components) depending on the value of the input parameter show . An HScene is initially not visible, a call to setVisible should be used to make it visible.

Hscene.setVisible(true) implicitly requests the user (input) focus for the HScene from the window manager -- however the granting of this request is dependent upon the window manager policy, etc.

Overrides:

java.awt.Component.setVisible(boolean) in class java.awt.Component

Parameters:

`show` - If true, makes this HScene visible; otherwise, hides this HScene .

org.havi.ui
HSceneFactory

Syntax

```
public class HSceneFactory

java.lang.Object
  |
  +--org.havi.ui.HSceneFactory
```

Description

The HSceneFactory class provides a generic mechanism for an application to request HScene resources from a (conceptual) window management system. The HSceneFactory is the single entry to potentially multiple GraphicsDevice centric window management policies.

The HSceneFactory class provides an opaque interface between any application (or window) management scheme and the Java application, itself.

Note that each application may acquire a maximum of one HScene , at any point in time. However, a new HScene may be acquired, provided that any previous HScene object has already been disposed. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Methods

dispose(HScene)

```
public void dispose(HScene scene)
```

An HScene may be terminated (ie its resources released to the system, for future garbage collection) by calling the dispose method on the HSceneFactory . After calling this method any further method calls on the HScene will result in java.lang.IllegalStateException being thrown.

This method allows the HSceneFactory to destroy an HScene .

Parameters:

`hscene` - the HScene to be destroyed.

See Also:

HScene

getBestScene(HSceneTemplate)

```
public HScene getBestScene(HSceneTemplate hst)
```

Returns an HScene that best corresponds to the input HSceneTemplate , or null if such an HScene cannot be generated.

Returns:
the HScene that matches the properties as specified in the HSceneTemplate , or null if they cannot be satisfied, or if no further HScene are available.

getBestSceneTemplate(HSceneTemplate)

```
public HSceneTemplate getBestSceneTemplate(HSceneTemplate hst)
```

Returns a HSceneTemplate that best corresponds to the input HSceneTemplate .

Note that since some platforms may support more than one concurrent application there is no guarantee that the values returned by this method would actually match those of a subsequently requested HScene , using the same template.

Parameters:
`hst` - The HSceneTemplate properties that the HScene should satisfy.

Returns:
an HSceneTemplate that best corresponds to the input HSceneTemplate .

getFullScreenScene(HGraphicsDevice, Dimension)

```
public HScene getFullScreenScene(HGraphicsDevice device, java.awt.Dimension resolution)
```

Create a full-screen HScene , at a specified pixel resolution.

Parameters:
`resolution` - the pixel resolution represented as a Dimension object.

Returns:
a created full-screen HScene , at a specified pixel resolution if possible, or null otherwise.

getInstance()

```
public static HSceneFactory getInstance()
```

Returns an HSceneFactory object to an application.

Returns:
an HSceneFactory object to an application. Note that repeated invocations of this method should return the same object (reference).

getSelectedScene(HGraphicsConfiguration[], HScreenRectangle, Dimension)

```
public HScene getSelectedScene(HGraphicsConfiguration[] selection,
            HScreenRectangle screenRectangle, java.awt.Dimension resolution)
```

Create a HScene that is aligned exactly to the area on-screen, from a limited set of HGraphicsConfiguration , for example, those compatible with video presentation.

Parameters:
`selection` - an array of HGraphicsConfiguration objects amongst which the selection should be made.

`screenRectangle` - an HScreenRectangle denoting an on-screen location.

`resolution` - the pixel resolution represented as a Dimension object.

Returns:

a created HScene , derived from a set of HGraphicsConfiguration objects, an on-screen location and a pixel resolution if possible, or null otherwise.

resizeScene(HScene, HSceneTemplate)

```
public HSceneTemplate resizeScene(HScene hs, HSceneTemplate hst)
```

Resizes a HScene so that it best corresponds to the input HSceneTemplate , or remains unchanged if it cannot be so resized.

Parameters:

`hs` - the HScene to be resized.

`hst` - the HSceneTemplate which denotes the new size / location. Only size / location options in the HSceneTemplate will be considered.

Returns:

an HSceneTemplate that indicates the HScene properties after (possible) resizing.

Throws:

`java.lang.IllegalStateException` - if the HScene had previously been disposed.

org.havi.ui
HSceneTemplate

Syntax

```
public class HSceneTemplate

java.lang.Object
  |
  +--org.havi.ui.HSceneTemplate
```

Description

The HSceneTemplate class is used to obtain an HScene subject to a variety of constraints. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

See Also:

HSceneFactory

Fields

GRAPHICS_CONFIGURATION

```
public static final int GRAPHICS_CONFIGURATION
```

A value for use in the preference field of the setPreference(int, Object, int) method in the HSceneTemplate that indicates that the HScene be created with a specified HGraphicsConfiguration (corresponding to a particular HGraphicsDevice).

By default the HSceneTemplate creates HScene on the default HScreen default HGraphicsDevice with its current HGraphicsConfiguration .

LARGEST_DIMENSION

```
public static final java.awt.Dimension LARGEST_DIMENSION
```

A Dimension object for use in the object field of the setPreference(int, Object, int) method in the HSceneTemplate that indicates that this feature should be returned in its largest possible dimension.

PREFERRED

```
public static final int PREFERRED
```

A value for use in the priority field of the setPreference(int, Object, int) method in the HSceneTemplate that indicates that this feature is preferred over a selection that does not include this feature, although both selections can be considered valid.

REQUIRED

```
public static final int REQUIRED
```

A value for use in the priority field of the setPreference(int, Object, int) method in the HSceneTemplate that indicates that this feature is required in the HScene . If this feature is not available, do not create an HScene object.

SCENE_PIXEL_RECTANGLE

```
public static final int SCENE_PIXEL_RECTANGLE
```

A value for use in the preference field of the setPreference(int, int) method in the HSceneTemplate that indicates that the HScene be created with a preferred location in pixels as given by a Rectangle object. The graphics pixels shall correspond to the pixel setting for the HGraphicsDevice settings as indicated by the HGraphicsConfiguration as specified in the HSceneTemplate (or its default value).

SCENE_PIXEL_RESOLUTION

```
public static final int SCENE_PIXEL_RESOLUTION
```

A value for use in the preference field of the setPreference(int, Object, int) method in the HSceneTemplate that indicates that the HScene be created with a preferred resolution in pixels as given by a Dimension object. If the Dimension object is LARGEST_DIMENSION then the returned HScene should have the greatest possible resolution.

SCENE_SCREEN_RECTANGLE

```
public static final int SCENE_SCREEN_RECTANGLE
```

A value for use in the preference field of the setPreference(int, int) method in the HSceneTemplate that indicates that the HScene be created with a preferred on- screen location as given by an HScreenRectangle object.

UNNECESSARY

```
public static final int UNNECESSARY
```

A value for use in the priority field of the setPreference(int, Object, int) method in the HSceneTemplate that indicates that this feature unnecessary in the HScene . A selection without this feature is preferred over a selection that includes this feature since it is not used.

Constructors

HSceneTemplate()

```
public HSceneTemplate()
```

Methods

getPreferenceObject(int)

```
public java.lang.Object getPreferenceObject(int preference)
```

Return the preference object for the specified preference.

Parameters:

`preference` - the preference to be indicated.

Returns:

the preference object for the specified preference. Valid values include:

- An HGraphicsConfiguration object which is returned for the GRAPHICS_CONFIGURATION preference.
- A java.awt.Rectangle object which is returned for the SCENE_PIXEL_RECTANGLE preference.
- A HScreenRectangle object which is returned for the SCENE_SCREEN_RECTANGLE preference.

getPreferencePriority(int)

```
public int getPreferencePriority(int preference)
```

Return the priority for the specified preference.

Parameters:

`preference` - the preference to be indicated. Valid values include: GRAPHICS_CONFIGURATION , SCENE_PIXEL_RECTANGLE and SCENE_SCREEN_RECTANGLE .

Returns:

the priority for the specified preference.

setPreference(int, Object, int)

```
public void setPreference(int preference, java.lang.Object object, int priority)
```

Set the indicated preference (and associated value object) to have the specified priority. If the preference has been previously set, then the previous object and priority shall be overwritten.

By default, the preferences should have an UNNECESSARY priority.

Parameters:

`preference` - the preference to be indicated. Valid values include: GRAPHICS_CONFIGURATION , SCENE_PIXEL_RECTANGLE and SCENE_SCREEN_RECTANGLE .

`object` - the Object associated with the given preference.

`priority` - the priority of the preference. Valid values include: REQUIRED , PREFERRED

org.havi.ui
HScreen

Syntax

```
public class HScreen

java.lang.Object
  |
  +--org.havi.ui.HScreen
```

Description

This class describes the final output composition of a device. It ties together all the (MPEG) video decoders, all the graphics sub-systems and backgrounds which are all combined together before finally being displayed. A platform with two independent displays would support two instances of this class. Where a device outputs audio closely bound with video, that audio output can also be represented through this class. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Methods

getCoherentScreenConfigurations(HScreenConfigTemplate[])

```
public HScreenConfiguration[] getCoherentScreenConfigurations(HScreenConfigTemplate[]
            hscta)
```

Return a coherent set of HScreenConfiguration matching a set of templates. One HScreenConfiguration will be returned for each HScreenConfigTemplate provided as input. The class of the returned objects will correspond to the class of the templates provided as input - where an HGraphicsConfigTemplate is provided as input, an HGraphicsConfiguration shall be returned. Where an HVideoConfigTemplate is provided as input, an HVideoConfiguration shall be returned. If more than one template of the same type is provided then the configurations returned must be on different devices but presented on the same screen.

Coherent means that all the required properties are respected in all of the templates provided and that a configuration can be returned for each template provided.

Parameters:

`hscta` - an array of objects describing desired / required configurations

Returns:

an array of non-null objects describing a coherent set of screen device configurations or null if no such coherent set is possible

getDefaultHBackgroundDevice()

```
public HBackgroundDevice getDefaultHBackgroundDevice()
```

Return the default background device for this screen.

Returns:
the default background device for this screen or null if none exist

getDefaultHGraphicsDevice()

```
public HGraphicsDevice getDefaultHGraphicsDevice()
```

Return the default graphics device for this screen. Note that the HGraphicsDevice is the default device for rendering graphics, but it may not be capable of displaying video / mixing it with graphics concurrently.

Returns:
the default graphics device for this screen or null if none exist

getDefaultHScreen()

```
public static HScreen getDefaultHScreen()
```

Returns the default HScreen for this application. For systems where an application is associated with audio or video which is started before the application starts, this method will return the HScreen where that associated audio / video is being output.

Returns:
the default HScreen for this application.

getDefaultHVideoDevice()

```
public HVideoDevice getDefaultHVideoDevice()
```

Return the default video device for this screen. Note that the HVideoDevice is the default device for rendering video, but it may not be capable of displaying graphics / mixing it with graphics concurrently.

Returns:
an HVideoDevice object or null if none exist

getHBackgroundDevices()

```
public HBackgroundDevice[] getHBackgroundDevices()
```

Returns a list of background devices for this screen.

Returns:
an array of HBackgroundDevice or null if none exist

getHGraphicsDevices()

```
public HGraphicsDevice[] getHGraphicsDevices()
```

Returns a list of graphics devices for this screen.

Returns:
an array of HGraphicsDevice or null if none exist

getHScreens()

```
public static HScreen[] getHScreens()
```

Returns all HScreen in this system.

Returns:

an array of HScreen representing all HScreen in this system.

getHVideoDevices()

```
public HVideoDevice[] getHVideoDevices()
```

Returns a list of video device for this screen. For systems where an application is associated with video started before the application starts, the first entry in the array returned will be the video device where that video is being output.

Returns:

an array of HVideoDevice objects or null if none exist.

setCoherentScreenConfigurations(HScreenConfiguration[])

```
public boolean setCoherentScreenConfigurations(HScreenConfiguration[] hsca)
```

Modify the settings for a set of HScreenDevice , based on their HScreenConfiguration supplied. Settings should be modified atomically (where possible) or should not be modified if the HScreenConfiguration can be determined to be conflicting a priori, i.e. are not "coherent", or would cause an exception to be thrown.

Parameters:

`hsca` - the array of configurations that should be applied atomically (where possible).

Returns:

A boolean indicating whether all HScreenConfiguration could be applied successfully. If all of the HScreenConfiguration could not be applied successfully, the configuration after this method may not match the configuration of the devices prior to this method being called --- applications should take steps to determine whether a partial change of settings has been made on each device.

Throws:

`java.lang.SecurityException` - if the application does not have sufficient rights to set the HScreenConfiguration for any of the devices.

HPermissionDeniedException - (HPermissionDeniedException) if the application does not currently have the right to set the configuration for any of the devices.

HConfigurationException - (HConfigurationException) if the specified HScreenConfiguration array is not valid for any of the devices.

org.havi.ui
HScreenConfigTemplate

Syntax
```
public abstract class HScreenConfigTemplate

java.lang.Object
  |
  +--org.havi.ui.HScreenConfigTemplate
```

Direct Known Subclasses:
HBackgroundConfigTemplate, HGraphicsConfigTemplate, HVideoConfigTemplate

Description
This class describes a configuration of a screen device in terms of various properties and their importance to the application. It is used to request a valid instance of a configuration conforming to the description provided. Sub-classes of this define additional constants which may be used for additional properties. Those classes can be sub-classed in turn to add further properties by systems using the HAVi UI.

Fields

FLICKER_FILTERING
```
public static final int FLICKER_FILTERING
```
A value for use in the preference field of the setPreference(int, int) method in the HScreenConfigTemplate that indicates that the device configuration supports flicker filtering (if it supports an interlaced screen).

INTERLACED_DISPLAY
```
public static final int INTERLACED_DISPLAY
```
A value for use in the preference field of the setPreference(int, int) method in the HScreenConfigTemplate that indicates that the device configuration supports an interlaced display.

PIXEL_ASPECT_RATIO
```
public static final int PIXEL_ASPECT_RATIO
```
A value for use in the preference field of the setPreference(int, int) method in the HScreenConfigTemplate that indicates that the device configuration supports the pixel aspect ratio, as specified in a Dimension object which indicates the (relative) x, y pixel aspect ratio.

If this preference (object) is not set, then by default the HScreenConfiguration should indicate pixel aspect ratio at some platform specific value, with PREFERRED priority.

PIXEL_RESOLUTION
```
public static final int PIXEL_RESOLUTION
```
A value for use in the preference field of the setPreference(int, int) method in the HScreenConfigTemplate that indicates that the device configuration supports the pixel resolution, as

specified in a Dimension object which indicates the pixel resolution of (the area of) the graphics device (as specified using the SCREEN_LOCATION preference).

If this preference (object) is not set, then by default the HScreenConfiguration should indicate pixel resolution at some platform specific value, with PREFERRED priority.

PREFERRED

```
public static final int PREFERRED
```

A value for use in the priority field of the setPreference(int, int) method in the HScreenConfigTemplate that indicates that this feature is desired in the HScreenConfiguration . A selection with this feature is preferred over a selection that does not include this feature, although both selections can be considered valid matches.

PREFERRED_NOT

```
public static final int PREFERRED_NOT
```

A value for use in the priority field of the setPreference(int, int) method in the HScreenConfigTemplate that indicates that this feature is desired not to be present in the HScreenConfiguration . A selection without this feature is preferred over a selection that does not include this feature, although both selections can be considered valid matches.

REQUIRED

```
public static final int REQUIRED
```

A value for use in the priority field of the setPreference(int, int) method in the HScreenConfigTemplate that indicates that this feature is required in the HScreenConfiguration . If this feature is not available, do not select the HScreenConfiguration object.

REQUIRED_NOT

```
public static final int REQUIRED_NOT
```

A value for use in the priority field of the setPreference(int, int) method in the HScreenConfigTemplate that indicates that this feature is required not to be present in the HScreenConfiguration . If this feature is available, do not select the HScreenConfiguration object.

SCREEN_LOCATION

```
public static final int SCREEN_LOCATION
```

A value for use in the preference field of the setPreference(int, int) method in the HScreenConfigTemplate that indicates that the device configuration supports graphics presentation on a particular on-screen area, as specified in an HScreenRectangle object

If this preference (object) is not set, then by default the HScreenConfiguration should indicate graphics presentation over the entire screen (only), with PREFERRED priority.

UNNECESSARY

```
public static final int UNNECESSARY
```

A value for use in the priority field of the setPreference(int, int) method in the HScreenConfigTemplate that indicates that this feature unnecessary in the HScreenConfiguration . A selection without this feature is preferred over a selection that includes this feature since it is not used.

VIDEO_GRAPHICS_PIXEL_ALIGNED

```
public static final int VIDEO_GRAPHICS_PIXEL_ALIGNED
```

A value for use in the preference field of the setPreference(int, int) method in the HScreenConfigTemplate that indicates that the device configuration supports the display of video streams and graphics with aligned pixels of the same size. Alignment of the origins of the two pixel coordinate spaces is explicitly not required. Where a video device is moving the video relative to the screen in real time (e.g. implementing pan and scan), graphics configurations shall only support this feature where the implementation of the graphics device can track the position changes in the video device automatically.

If this preference is set and used to request an HVideoConfiguration then an HGraphicsConfiguration shall be included in the template. If it is set and used to request an HGraphicsConfiguration then an HVideoConfiguration shall be included in the template. Requesting an HVideoConfiguration which is VIDEO_GRAPHICS_PIXEL_ALIGNED with another HVideoConfiguration shall fail unless the system concerned supports two HVideoDevice objects which can support this exact feature.

ZERO_GRAPHICS_IMPACT

```
public static final int ZERO_GRAPHICS_IMPACT
```

A value for use in the preference field of the setPreference(int, int) method in the HScreenConfigTemplate that indicates that the device configuration should have zero impact on already running graphical applications. If used with the REQUIRED priority, this means no changes shall be made. If used with the PREFERRED priority, this means changes may be made but should be minimised.

The PREFERRED_NOT and REQUIRED_NOT . priorities may be ignored in the selection of an HScreenConfiguration for this preference type.

ZERO_VIDEO_IMPACT

```
public static final int ZERO_VIDEO_IMPACT
```

A value for use in the preference field of the setPreference(int, int) method in the HScreenConfigTemplate that indicates that the device configuration should have zero impact on already running video streams. If used with the REQUIRED priority, this means no changes shall be made. If used with the PREFERRED priority, this means changes may be made but should be minimised.

The PREFERRED_NOT and REQUIRED_NOT . priorities may be ignored in the selection of an HScreenConfiguration for this preference type.

Constructors

HScreenConfigTemplate()

```
public HScreenConfigTemplate()
```

Constructor for an empty template

Methods

getPreferenceObject(int)

```
public java.lang.Object getPreferenceObject(int preference)
```

Return the preference object for the specified preference.

Parameters:
`preference` - the preference to be indicated. Valid values include: PIXEL_ASPECT_RATIO , PIXEL_RESOLUTION and SCREEN_LOCATION .

Returns:
the preference object for the specified preference.

getPreferencePriority(int)

```
public int getPreferencePriority(int preference)
```

Return the priority for the specified preference.

Parameters:
`preference` - the preference to be indicated. Valid values include: ZERO_GRAPHICS_IMPACT , ZERO_VIDEO_IMPACT , INTERLACED_DISPLAY , FLICKER_FILTERING , VIDEO_GRAPHICS_PIXEL_ALIGNED . PIXEL_ASPECT_RATIO , PIXEL_RESOLUTION and SCREEN_LOCATION .

Returns:
the priority for the specified preference.

isDisplayConfigSupported(HScreenConfiguration)

```
public boolean isDisplayConfigSupported(HScreenConfiguration hsc)
```

Returns a boolean indicating whether or not the specified HScreenConfiguration can be used to create a drawing surface that supports the indicated features.

Parameters:
`hsc` - - the HScreenConfiguration to test

Returns:
true if this HScreenConfiguration object can be used to create configurations that support the indicated features; false if the it can not be used to create a configuration as requested.

setPreference(int, int)

```
public void setPreference(int preference, int priority)
```

Set the indicated preference to have the specified priority. If the preference has been previously set, then the previous object and priority shall be overwritten.

By default the preferences should have an UNNECESSARY priority.

Parameters:
`preference` - the preference to be indicated. Valid values include: ZERO_GRAPHICS_IMPACT , ZERO_VIDEO_IMPACT , INTERLACED_DISPLAY , FLICKER_FILTERING and VIDEO_GRAPHICS_PIXEL_ALIGNED .

`priority` - the priority of the preference. Valid values include: REQUIRED , PREFERRED , UNNECESSARY , PREFERRED_NOT and REQUIRED_NOT .

setPreference(int, Object, int)

```
public void setPreference(int preference, java.lang.Object object, int priority)
```

Set the indicated preference (and associated value object) to have the specified priority. If the preference has been previously set, then the previous object and priority shall be overwritten.

By default the preferences should have an UNNECESSARY priority.

Parameters:
`preference` - the preference to be indicated. Valid values include: PIXEL_ASPECT_RATIO , PIXEL_RESOLUTION and SCREEN_LOCATION .

`object` - the Object associated with the given preference.

`priority` - the priority of the preference. Valid values include: REQUIRED , PREFERRED , UNNECESSARY , PREFERRED_NOT and REQUIRED_NOT .

org.havi.ui
HScreenConfiguration

Syntax
```
public abstract class HScreenConfiguration

java.lang.Object
  |
  +--org.havi.ui.HScreenConfiguration
```

Direct Known Subclasses:
HBackgroundConfiguration, HGraphicsConfiguration, HVideoConfiguration

Description
The HScreenConfiguration class describes the characteristics (settings) of an HScreenDevice . There can be many HScreenConfiguration objects associated with a single HScreenDevice .

See Also:
HScreenDevice

Methods

convertTo(HScreenConfiguration, Dimension)

```
public java.awt.Dimension convertTo(HScreenConfiguration destination, java.awt.
            Dimension source)
```

Convert a pixel position from one coordinate system to another without including any rounding errors from passing throgh normalised coordinates. This returns null if this transformation isn't possible for various reasons. These reasons include:

- at least one of the two HScreenConfiguration isn't pixel based or doesn't yet have a fixed location on the HScreen.
- a non-linear transformation is in use between the two.
- the information needed to calculate this isn't available.
- the transformation is changing with time (e.g. due to pan & scan).

Parameters:

`destination` - the destination HScreenConfiguration .

`source` - the pixel position in this HScreenConfiguration .

Returns:

the position of the specified pixel position measured in the destination coordinate system, or null if this isn't possible.

getFlickerFilter()

```
public boolean getFlickerFilter()
```

Return whether this configuration includes filtering to reduce interlace flicker.

Returns:

true if filtering is included

getInterlaced()

```
public boolean getInterlaced()
```

Return whether this configuration is interlaced

Returns:
true if this configuration is interlaced

getOffset(HScreenConfiguration)

```
public java.awt.Dimension getOffset(HScreenConfiguration hsc)
```

Returns the offset between the origin of the pixel coordinate space of the specified HScreenConfiguration , and the origin of the current pixel coordinate space of this HScreenConfiguration . The offset is returned in the pixel coordinate space of this HScreenConfiguration .

Parameters:
`hsc` - the HScreenConfiguration to which the offset between pixel origins should be recovered.

Returns:
the offset between the pixel coordinate space of the specified HScreenConfiguration and the current pixel coordinate space of this HScreenConfiguration . A null object may be returned if there is insufficient information to recover the pixel offset.

getPixelAspectRatio()

```
public java.awt.Dimension getPixelAspectRatio()
```

Return the pixel aspect ratio of this configuration. Some examples are {4,3}, {16:9}, {1:1}.

Returns:
the aspect ratio of the pixels in this configuration.

getPixelResolution()

```
public java.awt.Dimension getPixelResolution()
```

Return the resolution of this configuration in pixels. The pixel coordinate system used is that of the device concerned.

Returns:
the resolution of this configuration in pixels.

getScreenArea()

```
public HScreenRectangle getScreenArea()
```

Return the position and size of this configuration on the screen in screen coordinates.

Returns:
the area on the screen of this configuration in screen coordinates.

isCompatibleConfiguration(HScreenConfigTemplate)

```
public boolean isCompatibleConfiguration(HScreenConfigTemplate hsct)
```

This method determines whether a given HScreenConfiguration is compatible with the specified HScreenConfigTemplate . This mechanism allows an application to determine whether the current configuration of an HScreenDevice is compatible with its requirements, without explicitly reconfiguring the HScreenDevice .

Parameters:
`gct` - the HScreenConfigTemplate against which the HScreenConfiguration should be compared.

Returns:

true if the HScreenConfiguration is compatible with the HScreenConfigTemplate sct, false otherwise.

org.havi.ui
HScreenDevice

Syntax

```
public class HScreenDevice implements org.davic.resources.ResourceProxy, org.davic.resources.
    ResourceServer

java.lang.Object
   |
   +--org.havi.ui.HScreenDevice
```

Direct Known Subclasses:

HBackgroundDevice, HGraphicsDevice, HVideoDevice

All Implemented Interfaces:

org.davic.resources.ResourceProxy, org.davic.resources.ResourceServer

Description

An instance of the HScreen class represents a single independent video output signal from a device. Devices with multiple independent video output signals should support multiple instances of this class. A video output signal is created by adding together the contributions from the devices represented by a number of objects inheriting from the HScreenDevice class. These can be HGraphicsDevice objects, HVideoDevice objects and HBackgroundDevice objects. A given HScreen may support any number of any of these objects as far as the API is concerned however some form of profiling may restrict this. In reality right now, one instance of each is all that may reasonably expected to be present.

Each HScreenDevice can have multiple settings (HScreenConfiguration) but only one "setting" (HScreenConfiguration) can be active at any point in time. The current configuration can be determined on the HScreenDevice subclasses using their specific getCurrentConfiguration methods. The current configuration can be modified on the HScreenDevice subclasses using their specific setCurrentConfiguration methods (assuming sufficient rights, etc.).

Applications may select the best of these configurations for them by creating an instance of HScreenConfigTemplate and populating that with a number preferences each with a priority. The implementation then matches this template against the range of possible configurations and attempts to find one which matches the template provided. Priorities REQUIRED and REQUIRED_NOT must be respected. If they cannot be respected then the method call shall fail and not return any configuration. Priorities PREFERRED and PREFERRED_NOT should be respected as much as possible. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Methods

addResourceStatusEventListener(ResourceStatusListener)

```
public void addResourceStatusEventListener(org.davic.resources.ResourceStatusListener
            listener)
```

Register a listener for events about changes in the state of the ownership of this device. * @param listener the object to be informed of state changes

Specified By:
org.davic.resources.ResourceServer.addResourceStatusEventListener(org.davic.resources.ResourceStatusListener) in interface org.davic.resources.ResourceServer

See Also:
HScreenDeviceReleasedEvent, HScreenDeviceReservedEvent

addScreenConfigurationListener(HScreenConfigurationListener)

```
public void addScreenConfigurationListener(HScreenConfigurationListener hscl)
```

Add a HScreenConfigurationListener to this device, which is notified whenever the device's configuration is modified.

Parameters:
`hscl` - the HScreenConfigurationListener to be added to this device.

addScreenConfigurationListener(HScreenConfigurationListener, HScreenConfigTemplate)

```
public void addScreenConfigurationListener(HScreenConfigurationListener hscl,
            HScreenConfigTemplate hsct)
```

Add a HScreenConfigurationListener to this device, which is notified when the device's configuration is further modified so that it is no longer compatible with the specified HScreenConfigTemplate .

Note that if the device configuration does not match the specified template, then the listener should be added and a HScreenConfigurationEvent immediately generated for the specified HScreenConfigurationListener .

Parameters:
`hscl` - the HScreenConfigurationListener to be added to this device.

`hsct` - the HScreenConfigTemplate which is to be used to determine compatibility with the device configuration.

getClient()

```
public org.davic.resources.ResourceClient getClient()
```

Return the object which represents the intended owner of the resource. This is as specified in the last call to the reserveDevice method.

Specified By:
org.davic.resources.ResourceProxy.getClient() in interface org.davic.resources.ResourceProxy

Returns:
a representation of the intended owner of the resource

getIDstring()

```
public java.lang.String getIDstring()
```

Returns the identification string associated with this HScreenDevice .

Returns:

an identification string

getScreenAspectRatio()

```
public java.awt.Dimension getScreenAspectRatio()
```

Return the aspect ratio of the screen as far as is known. i.e. 4:3, 16:9, etc.

This Dimension may be used to determine the pixel aspect ratio for given HScreenConfiguration .

Returns:
a Dimension object specifying the aspect ratio of the screen

releaseDevice()

```
public void releaseDevice()
```

Release the ownership of this device. If this application doesn't own the device then this method has no effect. It is not specified whether any device configuration set by this application will be removed from display immediately or whether it will remain on display until a subsequent application obtains the device and sets its own configuration. Applications wishing to ensure a configuration they have installed is removed must actively remove it before calling this method.

removeResourceStatusEventListener(ResourceStatusListener)

```
public void removeResourceStatusEventListener(org.davic.resources.ResourceStatusListener
            listener)
```

Remove a listener for events about changes in the state of the ownership of this device. This method has no effect if the listener specified is not registered.

Specified By:
org.davic.resources.ResourceServer.removeResourceStatusEventListener(org.davic.resources.ResourceStatusListener) in interface org.davic.resources.ResourceServer

Parameters:
`listener` - the object which is no longer interested

See Also:
HScreenDeviceReleasedEvent, HScreenDeviceReservedEvent

removeScreenConfigurationListener(HScreenConfigurationListener)

```
public void removeScreenConfigurationListener(HScreenConfigurationListener hscl)
```

Remove a HScreenConfigurationListener from this device. if the specified listener is not registered, the method has no effect

Parameters:
`hscl` - the HScreenConfigurationListener to be removed from this device.

reserveDevice(ResourceClient)

```
public boolean reserveDevice(org.davic.resources.ResourceClient client)
```

Request the right to control this screen device.

Parameters:
`client` - a representation of the intended owner of the resource

Returns:
true if the right is granted otherwise false

org.havi.ui
HScreenPoint

Syntax

```
public class HScreenPoint

java.lang.Object
  |
  +--org.havi.ui.HScreenPoint
```

Description

HScreenPoint denotes a screen location expressed as a relative value of the screen dimensions. Note that since these are relative dimensions they are effectively independent of any particular screen's physical dimensions, or aspect ratio.

The x coordinate is in terms of the ratio of the particular horizontal screen location to the entire screen width.

The y coordinate is in terms of the ratio of the particular vertical screen location to the entire screen width.

All measurements should be taken from the top, left corner of the screen, measuring positive dimensions down and to the right.

Note that x and y coordinates are not constrained - they may be negative, or have values greater than one - and hence, may denote locations that are not "on- screen".

Hence,

- (0.0, 0.0) denotes the top, left hand corner of the screen.
- (1.0, 0.0) denotes the top, right hand corner of the screen.
- (0.5, 0.5) denotes the centre (middle) of the screen.
- (0.0, 1.0) denotes the bottom, left hand corner of the screen.
- (1.0, 1.0) denotes the bottom, right hand corner of the screen.

Note that in practice, particularly in the case of television, the precise location may vary slightly due to effects of overscan, etc. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

See Also:

HScreenRectangle

Fields

x

```
public float x
```

y

```
public float y
```

Constructors

HScreenPoint(float, float)

```
public HScreenPoint(float x, float y)
```

Methods

setLocation(float, float)

```
public void setLocation(float x, float y)
```

org.havi.ui
HScreenRectangle

Syntax

```
public class HScreenRectangle

java.lang.Object
  |
  +--org.havi.ui.HScreenRectangle
```

Description

HScreenRectangle denotes a screen area expressed as a relative value of the screen dimensions. Note that since these are relative dimensions they are effectively independent of any particular screen's physical dimensions, or aspect ratio.

Note that the x and y offset coordinates of the top, left corner of the area are not constrained - they may be negative, or have values greater than one - and hence, may denote an offset location that is not "on-screen". The width and height of the area should be positive (including zero), but are otherwise unconstrained - and hence may denote areas greater in size than the entire screen.

Hence,

- (0.0, 0.0, 1.0, 1.0) denotes the whole of the screen.
- (0.0, 0.0, 0.5, 0.5) denotes the top, left hand quarter of the screen.
- (0.5, 0.0, 0.5, 0.5) denotes the top, right hand quarter of the screen.
- (0.25, 0.25, 0.5, 0.5) denotes a centred quarter-screen area of the screen.
- (0.0, 0.5, 0.5, 0.5) denotes the bottom, left hand quarter of the screen.
- (0.5, 0.5, 0.5, 0.5) denotes the bottom, right hand quarter of the screen.

Note that in practice, particularly in the case of television, the precise location may vary slightly due to effects of overscan, etc.

Note that systems using HScreenRectangle directly should consider the effects of rounding errors, etc. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

See Also:

HScreenPoint

Fields

height

```
public float height
```

width

```
public float width
```

x

```
public float x
```

y

```
public float y
```

Constructors

HScreenRectangle(float, float, float, float)

```
public HScreenRectangle(float x, float y, float width, float height)
```

Methods

setLocation(float, float)

```
public void setLocation(float x, float y)
```

setSize(float, float)

```
public void setSize(float width, float height)
```

org.havi.ui
HSinglelineEntry

Syntax

```
public class HSinglelineEntry extends HVisible implements HValue, HKeyboardInputPreferred

java.lang.Object
  |
  +--java.awt.Component
        |
        +--HComponent
              |
              +--HVisible
                    |
                    +--org.havi.ui.HSinglelineEntry
```

Direct Known Subclasses:

HMultilineEntry

All Implemented Interfaces:

HKeyboardInputPreferred, HMatteLayer, HNavigable, HState, HValue, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

The HSinglelineEntry class is used to receive a single line of alphanumeric entry from the user and can also be used for password input. Upon creation the HSinglelineEntry widget is set to a non- editable mode identical in functionality to an HText .

On keyboard-based systems, if the user navigates to the widget using the keyboard then the widget must first be switched into edit mode before it will accept any key presses other than for navigation. The mechanism by which the widget is switched into and out of edit mode is via the HUIEvent mechanism or other java.awt techniques.

On entering its editable mode the widget will generate a VK_STARTCHANGE event. All java.awt key events will then be received by the HSinglelineEntry . On leaving its editable mode the widget must generate a VK_ENDCHANGE enent. The user can then navigate out of the HSinglelineEntry .

On mouse-based systems, if the user selects the widget with a mouse button then the HSinglelineEntry will be switched into edit mode and will stay in edit mode so long as the mouse pointer remains within the bounds of the widget. Once the mouse pointer leaves the bounds then it will switch back into non-editable mode. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Parameter	Description	Default value	Set method	Get method
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	Width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
height	Height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
text	The text within this HSinglelineEntry , to be used as the displayed and editable content, for both the NORMAL_STATE and FOCUS_STATE states.	null	setTextContent(String, int)	getTextContent(int)
maxChars	The maximum number of characters (per line) allowed in this HSinglelineEntry .	16 characters	setMaxCharsPerLine(int)	getMaxCharsPerLine()
font	The font to be used for this component.	---	java.awt.Component#setFont.	java.awt.Component#getFont.
color	The color to be used for this component.	---	java.awt.Component#setForeground.	java.awt.Component#getForeground.

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The default "look" for this class.	A platform specific HSinglelineEntryLook	setDefaultLook(HSinglelineEntryLook)	getDefaultLook()
The "look" for this object.	The HSinglelineEntryLook returned from HSinglelineEntry.getDefaultLook when this object was created.	setLook(HLook)	getLook()
The gain focus sound.	null	setGainFocusSound(HSound)	getGainFocusSound()

Description	Default value	Set method	Get method
The lose focus sound.	null	setLoseFocusSound(HSound)	getLoseFocusSound()
Caret position	At the end of the current text string	setCaretCharPosition(int)	getCaretCharPosition()
Input type	INPUT_ANY	setType(int)	getType()
Password protection (the echo character)	Entry is "clear", i.e. not password protected.	setEchoChar(char)	getEchoChar() / echoCharIsSet()

Fields

INPUT_ALPHANUMERIC

```
public static final int INPUT_ALPHANUMERIC
```

Indicates that the entry field should accept only letters and numbers, as determined by the java.lang. Character isLetterOrDigit method.

INPUT_ANY

```
public static final int INPUT_ANY
```

Indicates that the entry field should accept any input characters, as determined by the java.lang. Character isAny method.

INPUT_NUMERIC

```
public static final int INPUT_NUMERIC
```

Indicates that the entry field should accept only numeric input, as determined by the java.lang.Character isDigit method.

Constructors

HSinglelineEntry()

```
public HSinglelineEntry()
```

Creates a HSinglelineEntry widget that uses the default look returned from the method getDefaultLook.

HSinglelineEntry(int)

```
public HSinglelineEntry(int maxChars)
```

Creates an HSinglelineEntry object. See the class description for details of constructor parameters and default values.

HSinglelineEntry(int, int, int, int, int)

```
public HSinglelineEntry(int x, int y, int width, int height, int maxChars)
```

Creates an HSinglelineEntry object. See the class description for details of constructor parameters and default values.

HSinglelineEntry(String, int, Font, Color)

```
public HSinglelineEntry(java.lang.String text, int maxChars, java.awt.Font font, java.awt.
            Color color)
```

Creates an HSinglelineEntry object. See the class description for details of constructor parameters and default values.

HSinglelineEntry(String, int, int, int, int, int, Font, Color)

```
public HSinglelineEntry(java.lang.String text, int x, int y, int width, int height,
            int maxChars, java.awt.Font font, java.awt.Color color)
```

Creates an HSinglelineEntry object. See the class description for details of constructor parameters and default values.

Methods

addChangeListener(HValueChangeListener)

```
public void addChangeListener(HValueChangeListener l)
```

Specified By:
addChangeListener(HValueChangeListener) in interface HValue

See Also:
HValue

caretNextCharacter()

```
public void caretNextCharacter()
```

Move the caret to the next character.

caretPreviousCharacter()

```
public void caretPreviousCharacter()
```

Move the caret to the previous character.

deleteNextChar()

```
public boolean deleteNextChar()
```

Delete a character forward of the current caret position.

Returns:
true if a character was deleted, false otherwise.

deletePreviousChar()

```
public boolean deletePreviousChar()
```

Delete a character behind the current caret position.

Returns:

true if a character was deleted, false otherwise.

echoCharIsSet()

```
public boolean echoCharIsSet()
```

Indicates if this component has an echo character set, ie if the echo character is non-zero.

getCaretCharPosition()

```
public int getCaretCharPosition()
```

Gets the position of the text insertion caret for this the current line in this text component. The valid values of the caret position are from 0 to the length of the string retrieved using getTextContent(int) , where 0 implies insertion as the first character (i.e. at the start of the string) and getTextContent(int) implies that further characters are to be appended onto the end of the string. Hence, the valid caret positions for the string "abc" of length 3, are 0, 1, 2 and 3 --- with caret locations as shown below:

```
0 "a" 1 "b" 2 "c" 3
```

Returns:
the position of the text insertion caret.

getChangeSound()

```
public HSound getChangeSound()
```

Specified By:
getChangeSound() in interface HValue

See Also:
HValue

getDefaultLook()

```
public static HSinglelineEntryLook getDefaultLook()
```

Returns the currently set default HLook for HSinglelineEntry components.

Returns:
The HLook that is used by default when creating a new HSinglelineEntry component.

getEchoChar()

```
public char getEchoChar()
```

Returns the character to be used for echoing.

getGainFocusSound()

```
public HSound getGainFocusSound()
```

Specified By:
getGainFocusSound() in interface HNavigable

See Also:
HNavigable

getLoseFocusSound()

```
public HSound getLoseFocusSound()
```

Specified By:
getLoseFocusSound() in interface HNavigable

See Also:

HNavigable

getMaxCharsPerLine()

```
public int getMaxCharsPerLine()
```

Get maximum number of characters per (single) line. The behaviour of the widget when the last character on a line is typed is implementation dependent.

getMove(int)

```
public HNavigable getMove(int keyCode)
```

Specified By:
getMove(int) in interface HNavigable

See Also:
HNavigable

getTextContent(int)

```
public java.lang.String getTextContent(int state)
```

Gets the text content used in this HSinglelineEntry .

Note that HSinglelineEntry widgets do not support separate pieces of textual content per state (as defined in HState) --- rather a single piece of content is defined for all its interaction states.

Overrides:
getTextContent(int) in class HVisible

See Also:
setTextContent(String, int)

getType()

```
public int getType()
```

This gets the type of keyboard entry: ANY, ALPHANUMERIC or NUMERIC.

insertChar(char)

```
public boolean insertChar(char c)
```

Insert a character at the current caret position, subject to the maximum number of input characters.

Returns:
true if the character was inserted, false otherwise.

isFocusTraversable()

```
public boolean isFocusTraversable()
```

By default classes implementing HNavigable are focus-traversable.

Overrides:
isFocusTraversable() in class HVisible

Returns:
true

See Also:
`java.awt.Component.isFocusTraversable()`

isSelected()

```
public boolean isSelected()
```

Specified By:
isSelected() in interface HNavigable

See Also:
HNavigable

processChangeEvent(HValueChangeEvent)

```
protected void processChangeEvent(HValueChangeEvent evt)
```

Processes HValueChangeEvent enabled for this object. Subclasses of classes implementing processChangeEvent should call super.processChangeEvent to ensure that the change event is handled appropriately.

Parameters:
`evt` - the event to be processed.

See Also:
HValue

processEvent(AWTEvent)

```
protected void processEvent(java.awt.AWTEvent evt)
```

It is a valid implementation option for HSinglelineEntry to override the protected processEvent method.

Overrides:
java.awt.Component.processEvent(java.awt.AWTEvent) in class java.awt.Component

See Also:
`java.awt.Component.processEvent(AWTEvent)`

processFocusEvent(FocusEvent)

```
protected void processFocusEvent(java.awt.event.FocusEvent evt)
```

It is a valid implementation option for HSinglelineEntry to override the protected processFocusEvent method.

Overrides:
java.awt.Component.processFocusEvent(java.awt.event.FocusEvent) in class java.awt.Component

See Also:
`java.awt.Component.processFocusEvent(FocusEvent)`

processKeyEvent(KeyEvent)

```
protected void processKeyEvent(java.awt.event.KeyEvent evt)
```

It is a valid implementation option for HSinglelineEntry to override the protected processKeyEvent method.

Overrides:
java.awt.Component.processKeyEvent(java.awt.event.KeyEvent) in class java.awt.Component

See Also:
`java.awt.Component.processKeyEvent(KeyEvent)`

removeChangeListener(HValueChangeListener)

```
public void removeChangeListener(HValueChangeListener l)
```

Specified By:
removeChangeListener(HValueChangeListener) in interface HValue

See Also:
HValue

requestFocus()

```
public void requestFocus()
```

Specified By:
requestFocus() in interface HNavigable

Overrides:
java.awt.Component.requestFocus() in class java.awt.Component

See Also:
`java.awt.Component.requestFocus()`

setCaretCharPosition(int)

```
public void setCaretCharPosition(int position)
```

Sets the position of the text insertion caret for this text component.

Parameters:
`position` - the position of the text insertion caret.

setChangeSound(HSound)

```
public void setChangeSound(HSound sound)
```

Specified By:
setChangeSound(HSound) in interface HValue

See Also:
HValue

setDefaultLook(HSinglelineEntryLook)

```
public static void setDefaultLook(HSinglelineEntryLook look)
```

Sets the default HLook for all HSinglelineEntry Components.

Parameters:
`look` - The HLook that will be used by default when creating a new HSinglelineEntry component.

Throws:
HInvalidLookException - If the HLook is not an HSinglelineEntryLook .

setEchoChar(char)

```
public void setEchoChar(char c)
```

Sets the number of character to echo for this component.

Parameters:
`c` - the character used to echo any input, e.g. if c == '*' a password- style input may be achieved. If c is zero, then all characters will be echoed on-screen, this is the default behaviour.

setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
            HNavigable right)
```

Specified By:
setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) in interface HNavigable

See Also:
HNavigable

setGainFocusSound(HSound)

```
public void setGainFocusSound(HSound sound)
```

Specified By:
setGainFocusSound(HSound) in interface HNavigable

See Also:
HNavigable

setLook(HLook)

```
public void setLook(HLook hlook)
```

Sets the HLook for this component.

Overrides:
setLook(HLook) in class HVisible

Parameters:
`hlook` - The HLook that is to be used for this component.

Throws:
HInvalidLookException - If the Look is not an HSinglelineEntryLook .

setLoseFocusSound(HSound)

```
public void setLoseFocusSound(HSound sound)
```

Specified By:
setLoseFocusSound(HSound) in interface HNavigable

See Also:
HNavigable

setMaxCharsPerLine(int)

```
public void setMaxCharsPerLine(int maxCol)
```

Set maximum number of characters per (single) line.

setMove(int, HNavigable)

```
public void setMove(int keyCode, HNavigable target)
```

Specified By:
setMove(int, HNavigable) in interface HNavigable

See Also:
HNavigable

setTextContent(String, int)

```
public void setTextContent(java.lang.String string, int state)
```

Sets the text content used in this HSinglelineEntry .

Note that HSinglelineEntry widgets do not support separate pieces of textual content per state (as defined in HState) --- rather a single piece of content is defined for all its interaction states.

Additionally, the setTextContent(String, int) method truncates the string according to the current maxChars setting. The setTextContent(String, int) method also sets the caret (insertion) position at the end of the (possibly truncated) string.

Overrides:
setTextContent(String, int) in class HVisible

Parameters:
`string` - The content. If the content is null, then any currently assigned content shall be removed for the specified state.

`state` - The state of the widget for which this content should be displayed. This parameter shall be ignored and considered to have the value ALL_STATES .

See Also:
getTextContent(int)

setType(int)

```
public void setType(int type)
```

This sets the type of keyboard entry: ANY, ALPHANUMERIC or NUMERIC.

org.havi.ui
HSinglelineEntryLook

Syntax

```
public class HSinglelineEntryLook implements HLook

java.lang.Object
  |
  +--org.havi.ui.HSinglelineEntryLook
```

Direct Known Subclasses:

HMultilineEntryLook

All Implemented Interfaces:

java.lang.Cloneable, HLook

Description

The HSinglelineEntryLook class is used by the HSinglelineEntry component to display the entering of text. This look will be provided by the platform and the exact way in which it is rendered will be platform dependant. However, the HSinglelineEntryLook associated with a given HVisible should use:

- java.awt.Component#getForeground() to determine the Color to render the rectangle.
- Border widths (as set by setHorizontalBorderSpacing / setVerticalBorderSpacing) of 2 pixels by default.

This is the default look that is used by HSinglelineEntry and its subclasses. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

See Also:

HSinglelineEntry

Constructors

HSinglelineEntryLook()

```
public HSinglelineEntryLook()
```

Creates an HSinglelineEntryLook that will be used for entering text.

Methods

getHorizontalBorderSpacing()

```
public int getHorizontalBorderSpacing()
```

Specified By:
getHorizontalBorderSpacing() in interface HLook

See Also:
HLook

getMaximumSize(HVisible)

```
public java.awt.Dimension getMaximumSize(HVisible visible)
```

Specified By:
getMaximumSize(HVisible) in interface HLook

See Also:
HLook

getMinimumSize(HVisible)

```
public java.awt.Dimension getMinimumSize(HVisible visible)
```

Specified By:
getMinimumSize(HVisible) in interface HLook

See Also:
HLook

getPreferredSize(HVisible)

```
public java.awt.Dimension getPreferredSize(HVisible visible)
```

Specified By:
getPreferredSize(HVisible) in interface HLook

See Also:
HLook

getVerticalBorderSpacing()

```
public int getVerticalBorderSpacing()
```

Specified By:
getVerticalBorderSpacing() in interface HLook

See Also:
HLook

setHorizontalBorderSpacing(int)

```
public void setHorizontalBorderSpacing(int width)
```

Specified By:
setHorizontalBorderSpacing(int) in interface HLook

See Also:
HLook

setVerticalBorderSpacing(int)

```
public void setVerticalBorderSpacing(int width)
```

Specified By:
setVerticalBorderSpacing(int) in interface HLook

See Also:
HLook

showLook(Graphics, HVisible, int)

```
public void showLook(java.awt.Graphics g, HVisible visible, int state)
```

Specified By:
showLook(Graphics, HVisible, int) in interface HLook

See Also:
HLook

org.havi.ui
HSound

Syntax
```
public class HSound

java.lang.Object
  |
  +--org.havi.ui.HSound
```

Description
The HSound class is used to represent an audio clip. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

None:

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The starting position of any audio clip to be played.	At the beginning of the audio clip.	---	---

Constructors

HSound()

```
public HSound()
```

Constructs a HSound object

Methods

dispose()

```
public void dispose()
```

If the HSound object is playing / looping then it will be stopped. The dispose method then discards all sample resources used by the HSound object. This mechanism returns the HSound object to the state before a load() method was invoked.

load(String)

```
public void load(java.lang.String location)
```

Loads data synchronously into an HSound object from an audio sample indicated by a locator.

Parameters:

`location` - is a String that encapsulates the location of an audio sample.

Throws:

`java.io.InterruptedIOException` - if the sample cannot be loaded from the location, due to an IO problem.

`SecurityException` - if the application does not have sufficient rights to load the sample from the specified location.

loop()

```
public void loop()
```

Starts the HSound class looping from the beginning of its associated audio data. If the sample data has not been completely loaded, this method has no effect.

When the audio data has been played in its entirety, then it should be played again from the beginning of its associated data, so as to cause a "seamless" continuous (infinite) audio playback - until the next stop , or play method is invoked. Note that the audio data is played back asynchronously, there is no mechanism for synchronisation with other classes presenting sounds, images, or video.

This method may fail "silently" if (local) audio facilities are unavailable on the platform.

play()

```
public void play()
```

Starts the HSound class playing from the beginning of its associated audio data. If the sample data has not been completely loaded, this method has no effect.

When the audio data has been played in its entirety, then no further audible output should be made, until the next play , or loop method is invoked. Note that the audio data is played back asynchronously, there is no mechanism for synchronisation with other classes presenting sounds, images, or video.

This method may fail "silently" if (local) audio facilities are unavailable on the platform.

stop()

```
public void stop()
```

Stops the HSound class playing its associated audio data.

Note that, if a play or loop method is invoked, after a stop , then presentation of the audio data will restart from the beginning of the audio data, rather than from the position where the audio data was stopped.

org.havi.ui
HState

Syntax
```
public interface HState
```

All Known Implementing Classes:
HVisible, HListGroup

Description
The HState interface encapsulates constants for widget states which are used in the HVisible setContent and getContent methods, to indicate which state the specified content is to be set.

See Also:
setTextContent(String, int), getTextContent(int), setGraphicContent(Image, int), getGraphicContent(int), setAnimateContent(Image[], int), getAnimateContent(int), setContent(Object, int), getContent(int)

Fields

ACTION_STATE

```
public static final int ACTION_STATE
```

This constant is applicable to all HActionable widgets and is used to indicate the actioned state (with focus).

See Also:
HActionable

ALL_STATES

```
public static final int ALL_STATES
```

Constant used to indicate all of the applicable states for a given widget.

Note that the ALL_STATES constant should only be used in setting content: setTextContent(String, int) setGraphicContent(Image, int) setAnimateContent(Image[], int) setContent(Object, int)

The ALL_STATES constant should not be used for retrieving content: getTextContent(int) getGraphicContent(int) getAnimateContent(int) getContent(int)

FIRST_STATE

```
public static final int FIRST_STATE
```

Constant used to indicate the value of the first (builtin) widget state.

FOCUS_STATE

```
public static final int FOCUS_STATE
```

This constant is applicable to all HNavigable widgets and is used to indicate the focused state.

See Also:

HNavigable

LAST_STATE

```
public static final int LAST_STATE
```

Constant used to indicate the value of the last (builtin) widget state.

NORMAL_ACTIONED_STATE

```
public static final int NORMAL_ACTIONED_STATE
```

This constant is applicable to all HSwitchable widgets and is used to indicate the actioned state (without focus).

See Also:
HSwitchable

NORMAL_STATE

```
public static final int NORMAL_STATE
```

This constant is applicable to all HVisible widgets and is used to indicate the normal, unfocused state.

See Also:
HVisible

org.havi.ui
HStaticAnimation

Syntax

```
public class HStaticAnimation extends HVisible implements HNoInputPreferred, HAnimateEffect
```

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--HComponent
              |
              +--HVisible
                    |
                    +--org.havi.ui.HStaticAnimation
```

Direct Known Subclasses:

HAnimation

All Implemented Interfaces:

HAnimateEffect, HMatteLayer, HNoInputPreferred, HState, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

The HStaticAnimation class is a user interface component used to display a sequence of images as an animation, but does *not* permit the user to navigate (focus) upon it.

The HStaticAnimation class supports animating images a finite number of times or infinitely (continuously), and either forward or in alternating directions with a specified time delay between the rendering of consecutive images. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	Width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Parameter	Description	Default value	Set method	Get method
height	Height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
images	The array of images to be used as the content for both the NORMAL_STATE state of this component.	null	setAnimateContent(Image[], int)	getAnimateContent(int)
delay	The delay between the presentation of successive content in the animation.	1 (i.e. 0.1 seconds)	setDelay(int)	getDelay()
repeatCount	The number of times that the animation is to be played.	REPEAT_INFINITE	setRepeatCount(int)	getRepeatCount()
playMode	The playing mode for the animation.	PLAY_REPEATING	setPlayMode(int)	getPlayMode()

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The default "look" for this class.	A platform specific HAnimateLook	setDefaultLook(HAnimateLook)	getDefaultLook()
The "look" for this object.	The HAnimateLook returned from HStaticAnimation.getDefaultLook when this object was created.	setLook(HLook)	getLook()
The intial piece of content to be presented, i.e. its position in the content array.	0	setPosition(int)	getPosition()
By default the animation should be stopped. Hence, to start the animation its start method must be explicitly invoked. This mechanism allows for animations that are programatically controlled, eg via the setPosition method.	"stopped"	start() / stop()	isAnimated()

Constructors

HStaticAnimation()

```
public HStaticAnimation()
```

Creates an HStaticAnimation object. See the class description for details of constructor parameters and default values.

HStaticAnimation(Image[], int, int, int)

```
public HStaticAnimation(java.awt.Image[] images, int delay, int playMode, int repeatCount)
```

Creates an HStaticAnimation object. See the class description for details of constructor parameters and default values.

HStaticAnimation(Image[], int, int, int, int, int, int, int)

```
public HStaticAnimation(java.awt.Image[] images, int delay, int playMode, int repeatCount,
            int x, int y, int width, int height)
```

Creates an HStaticAnimation object. See the class description for details of constructor parameters and default values.

Methods

getDefaultLook()

```
public static HAnimateLook getDefaultLook()
```

Returns the currently set default HLook for HStaticAnimation components.

Returns:
The HLook that is used by default when creating a new HStaticAnimation component.

getDelay()

```
public int getDelay()
```

Specified By:
getDelay() in interface HAnimateEffect

See Also:
HAnimateEffect

getPlayMode()

```
public int getPlayMode()
```

Specified By:
getPlayMode() in interface HAnimateEffect

See Also:
HAnimateEffect

getPosition()

```
public int getPosition()
```

Specified By:
getPosition() in interface HAnimateEffect

See Also:
HAnimateEffect

getRepeatCount()

```
public int getRepeatCount()
```

Specified By:
getRepeatCount() in interface HAnimateEffect

See Also:
HAnimateEffect

isAnimated()

```
public boolean isAnimated()
```

Specified By:
isAnimated() in interface HAnimateEffect

See Also:
HAnimateEffect

setDefaultLook(HAnimateLook)

```
public static void setDefaultLook(HAnimateLook hlook)
```

Sets the default HLook for all HStaticAnimation Components.

Parameters:
`hlook` - The HLook that will be used by default when creating a new HStaticAnimation component.

Throws:
HInvalidLookException - If the HLook is not an HAnimateLook .

setDelay(int)

```
public void setDelay(int count)
```

Specified By:
setDelay(int) in interface HAnimateEffect

See Also:
HAnimateEffect

setLook(HLook)

```
public void setLook(HLook hlook)
```

Sets the HLook for this component.

Overrides:
setLook(HLook) in class HVisible

Parameters:
`hlook` - The HLook that is to be used for this component.

Throws:
HInvalidLookException - If the HLook is not an HAnimateLook .

setPlayMode(int)

```
public void setPlayMode(int mode)
```

Specified By:

setPlayMode(int) in interface HAnimateEffect

See Also:
HAnimateEffect

setPosition(int)

```
public void setPosition(int position)
```

Specified By:
setPosition(int) in interface HAnimateEffect

See Also:
HAnimateEffect

setRepeatCount(int)

```
public void setRepeatCount(int count)
```

Specified By:
setRepeatCount(int) in interface HAnimateEffect

See Also:
HAnimateEffect

start()

```
public void start()
```

Specified By:
start() in interface HAnimateEffect

See Also:
HAnimateEffect

stop()

```
public void stop()
```

Specified By:
stop() in interface HAnimateEffect

See Also:
HAnimateEffect

org.havi.ui
HStaticIcon

Syntax

```
public class HStaticIcon extends HVisible implements HNoInputPreferred
```

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--HComponent
              |
              +--HVisible
                    |
                    +--org.havi.ui.HStaticIcon
```

Direct Known Subclasses:

HIcon

All Implemented Interfaces:

HMatteLayer, HNoInputPreferred, HState, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

This class creates an HStaticIcon (a graphical image). By default it uses the HGraphicLook to render itself. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	Width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Parameter	Description	Default value	Set method	Get method
height	Height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
image	The image to be used as the content for the NORMAL_STATE , state of this component.	null	setGraphicContent(Image, int)	getGraphicContent(int)

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The default "look" for this class.	A platform specific HGraphicLook	setDefaultLook(HGraphicLook)	getDefaultLook()
The "look" for this object.	The HGraphicLook returned from getDefaultLook() when this object was created.	setLook(HLook)	getLook()

Constructors

HStaticIcon()

```
public HStaticIcon()
```

Creates an HStaticIcon object. See the class description for details of constructor parameters and default values.

HStaticIcon(Image)

```
public HStaticIcon(java.awt.Image image)
```

Creates an HStaticIcon object. See the class description for details of constructor parameters and default values.

HStaticIcon(Image, int, int, int, int)

```
public HStaticIcon(java.awt.Image image, int x, int y, int width, int height)
```

Creates an HStaticIcon object. See the class description for details of constructor parameters and default values.

Methods

getDefaultLook()

```
public static HGraphicLook getDefaultLook()
```

Returns the currently set default HLook for HStaticIcon components.

Returns:
The HLook that is used by default when creating a new HStaticIcon component.

setDefaultLook(HGraphicLook)

```
public static void setDefaultLook(HGraphicLook hlook)
```

Sets the default HLook for all HStaticIcon Components.

Parameters:
`hlook` - The HLook that will be used by default when creating a new HStaticIcon component.

Throws:
HInvalidLookException - If the HLook is not an HGraphicLook .

setLook(HLook)

```
public void setLook(HLook hlook)
```

Sets the HLook for this component.

Overrides:
setLook(HLook) in class HVisible

Parameters:
`hlook` - The HLook that is to be used for this component.

Throws:
HInvalidLookException - If the HLook is not an HGraphicLook .

org.havi.ui
HStaticRange

Syntax

```
public class HStaticRange extends HVisible implements HNoInputPreferred
```

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--HComponent
              |
              +--HVisible
                    |
                    +--org.havi.ui.HStaticRange
```

Direct Known Subclasses:

HRange

All Implemented Interfaces:

HMatteLayer, HNoInputPreferred, HState, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

The HStaticRange component is used for displaying a value which is within a fixed range. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	Width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Parameter	Description	Default value	Set method	Get method
height	Height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
orientation	The "orientation" of the range object.	OR_HORIZ	setOrientation(int)	getOrientation()
minimum	The minimum value that can be returned by this range object.	0	setRange(int, int)	getMinValue()
maximum	The maximum value that can be returned by this range object.	100	setRange(int, int)	getMaxValue()
value	The current value returned by this range object.	0	setValue(int)	getValue()

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The default "look" for this class.	A platform specific HRangeLook	setDefaultLook(HRangeLook)	getDefaultLook()
The "look" for this object.	The HRangeLook returned from HStaticRange.getDefaultLook when this object was created.	setLook(HLook)	getLook()
The offsets for the "thumb" of this range control	min = 0, max = 0	setThumbOffsets(int, int)	getThumbMinOffset() / getThumbMaxOffset()
The behavior of this range object with respect to its "thumb" values	SLIDER_BEHAVIOR	setBehavior(int)	getBehavior()

Fields

OR_DIAL

```
public static final int OR_DIAL
```

The HStaticRange should be rendered with a dial "orientation".

OR_HORIZ

```
public static final int OR_HORIZ
```

The HStaticRange should be rendered with a horizontal orientation.

OR_VERT

```
public static final int OR_VERT
```

The HStaticRange should be rendered with a vertical orientation.

SCROLLBAR_BEHAVIOR

```
public static final int SCROLLBAR_BEHAVIOR
```

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

SLIDER_BEHAVIOR

```
public static final int SLIDER_BEHAVIOR
```

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

Constructors

HStaticRange()

```
public HStaticRange()
```

Creates an HStaticRange object. See the class description for details of constructor parameters and default values.

HStaticRange(int, int, int, int)

```
public HStaticRange(int orientation, int minimum, int maximum, int value)
```

Creates an HStaticRange object. See the class description for details of constructor parameters and default values.

HStaticRange(int, int, int, int, int, int, int, int)

```
public HStaticRange(int orientation, int minimum, int maximum, int value, int x, int y,
            int width, int height)
```

Creates an HStaticRange object. See the class description for details of constructor parameters and default values.

Methods

getBehavior()

```
public int getBehavior()
```

Returns the behavior for this HStaticRange .

Returns:
the behavior for this HStaticRange .

getDefaultLook()

```
public static HRangeLook getDefaultLook()
```

Returns the currently set default HLook for HStaticRange components.

Returns:
The HLook that is used by default when creating a new HStaticRange component.

getMaxValue()

```
public int getMaxValue()
```

Get the maximum value of the range

Returns:
The maximum value of the range.

getMinValue()

```
public int getMinValue()
```

Gets the minimum of the range.

Returns:
The minimum value for the range

getOrientation()

```
public int getOrientation()
```

Gets the orientation of the range.

Returns:
the orientation, OR_HORIZ , OR_VERT or OR_DIAL .

getThumbMaxOffset()

```
public int getThumbMaxOffset()
```

Returns the thumb offset for its maximum value.

Returns:
the thumb offset for its maximum value.

getThumbMinOffset()

```
public int getThumbMinOffset()
```

Returns the thumb offset for its minimum value.

Returns:
the thumb offset for its minimum value.

getValue()

```
public int getValue()
```

Gets the value of the control. Note that the recovered value is subject to the control's current behavior.

See Also:
SLIDER_BEHAVIOR, SCROLLBAR_BEHAVIOR

setBehavior(int)

```
public void setBehavior(int behavior)
```

Sets the behavior for this HStaticRange .

Parameters:
`behavior` - the behavior for this HStaticRange (SLIDER_BEHAVIOR or SCROLLBAR_BEHAVIOR).

setDefaultLook(HRangeLook)

```
public static void setDefaultLook(HRangeLook look)
```

Sets the default HLook for all HStaticRange Components.

Parameters:
`look` - The HLook that will be used by default when creating a new HStaticRange component.

Throws:
HInvalidLookException - If the HLook is not an HRangeLook .

setLook(HLook)

```
public void setLook(HLook hlook)
```

Sets the HLook for this component.

Overrides:
setLook(HLook) in class HVisible

Parameters:
`hlook` - The HLook that is to be used for this component.

Throws:
HInvalidLookException - If the Look is not an HRangeLook .

setOrientation(int)

```
public void setOrientation(int orientation)
```

Sets the orientation of the range.

Parameters:
`orientation,` - OR_HORIZ , OR_VERT or OR_DIAL .

setRange(int, int)

```
public boolean setRange(int minimum, int maximum)
```

Sets the range of values for the control.

Parameters:
`minimum` - The minimum value of the range control

`maximum` - The maximum value of the range control

Returns:
Indicates if the min and max values have been set correctly. Returns false if the minimum value is greater than or equal to the maximum value, otherwise returns true

setThumbOffsets(int, int)

```
public void setThumbOffsets(int minOffset, int maxOffset)
```

Set the offsets for the "thumb" area on this range control. The "thumb" is then drawn from (value - minOffset), to (value + maxOffset) positions outside of the HRange values [minimum:maximum] are clipped to the closest value.

There is no requirement that minOffset == maxOffset. Both offsets may be zero, yielding a thermometer-like range object. All measurements are in the same units as the minimum / maximum values on

the HStaticRange object. The size of the "thumb" is the application author's responsibility. By default the "thumb" does not affect the range over which the value of the HStaticRange may be modified. It is recommended that the HRangeLook provides mechanisms to denote the value of the HStaticRange , in addition to indicating the extent of the thumb as defined by the offsets.

See Also:
setBehavior(int)

setValue(int)

```
public void setValue(int value)
```

Sets the value of the control, subject to its current behavior.

Parameters:
`value` - the value to which the control should be set.

See Also:
SLIDER_BEHAVIOR, SCROLLBAR_BEHAVIOR

org.havi.ui
HStaticText

Syntax

```
public class HStaticText extends HVisible implements HNoInputPreferred
```

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--HComponent
              |
              +--HVisible
                    |
                    +--org.havi.ui.HStaticText
```

Direct Known Subclasses:

HText

All Implemented Interfaces:

HMatteLayer, HNoInputPreferred, HState, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

The HStaticText class is used to define a piece of textual content (label) that cannot be navigated to. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	Width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Parameter	Description	Default value	Set method	Get method
height	Height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
text	The string to be used as the content for the NORMAL_STATE state of this component.	null	setTextContent(String, int)	getTextContent(int)
tlm	The text layout manager responsible for text formatting.	An HDefaultTextLayoutManager object.	setTextLayoutManager(HTextLayoutManager)	getTextLayoutManager()
font	The font for this component.	---	java.awt.Component#setFont	java.awt.Component#getFont
background	The background color for this component.	---	java.awt.Component#getBackground	java.awt.Component#setBackground
foreground	The foreground color for this component.	---	java.awt.Component#getForeground	java.awt.Component#setForeground

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The default "look" for this class.	A platform specific HTextLook	setDefaultLook(HTextLook)	getDefaultLook()
The "look" for this object.	The HTextLook returned from getDefaultLook() when this object was created.	setLook(HLook)	getLook()

Constructors

HStaticText()

```
public HStaticText()
```

Creates an HStaticText object. See the class description for details of constructor parameters and default values.

HStaticText(String)

```
public HStaticText(java.lang.String text)
```

Creates an HStaticText object. See the class description for details of constructor parameters and default values.

HStaticText(String, Font, Color, Color, HTextLayoutManager)

```
public HStaticText(java.lang.String text, java.awt.Font font, java.awt.Color foreground,
            java.awt.Color background, HTextLayoutManager tlm)
```

Creates an HStaticText object. See the class description for details of constructor parameters and default values.

HStaticText(String, int, int, int, int)

```
public HStaticText(java.lang.String text, int x, int y, int width, int height)
```

Creates an HStaticText object. See the class description for details of constructor parameters and default values.

HStaticText(String, int, int, int, int, Font, Color, Color, HTextLayoutManager)

```
public HStaticText(java.lang.String text, int x, int y, int width, int height, java.awt.
            Font font, java.awt.Color foreground, java.awt.Color background,
            HTextLayoutManager tlm)
```

Creates an HStaticText object. See the class description for details of constructor parameters and default values.

Methods

getDefaultLook()

```
public static HTextLook getDefaultLook()
```

Returns the currently set default HLook for HStaticText components.

Returns:
The HLook that is used by default when creating a new HStaticText component.

setDefaultLook(HTextLook)

```
public static void setDefaultLook(HTextLook hlook)
```

Sets the default HLook for all HStaticText Components.

Parameters:
`hlook` - The HLook that will be used by default when creating a new HStaticText component.

Throws:
HInvalidLookException - If the HLook is not an HTextLook .

setLook(HLook)

```
public void setLook(HLook hlook)
```

Sets the HLook for this component.

Overrides:
setLook(HLook) in class HVisible

Parameters:
`hlook` - The HLook that is to be used for this component.

Throws:
HInvalidLookException - If the Look is not an HTextLook .

org.havi.ui

HStillImageBackgroundConfiguration

Syntax

```
public class HStillImageBackgroundConfiguration extends HBackgroundConfiguration
```

```
java.lang.Object
  |
  +--HScreenConfiguration
        |
        +--HBackgroundConfiguration
              |
              +--org.havi.ui.HStillImageBackgroundConfiguration
```

Description

This class represents a background configuration which supports the installation of still images. The platform using the HAVi user-interface specification must specify which image formats are supported. The java.awt.Image class is intentionally not used in order to allow the support of image formats which carry sufficent restrictions that expressing them through the API of that class would require extensive use of runtime errors. One specific example of this is MPEG I-frames. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

None.

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Constructors

HStillImageBackgroundConfiguration()

```
protected HStillImageBackgroundConfiguration()
```

It is not intended that applications should directly construct HStillImageBackgroundConfiguration objects. Creates an HStillImageBackgroundConfiguration object. See the class description for details of constructor parameters and default values.

Methods

displayImage(HBackgroundImage)

```
public void displayImage(HBackgroundImage image)
```

Display an image. If the data for the image has not been loaded then this method will block while the data is loaded.

Parameters:

`image` - the image to display

displayImage(HBackgroundImage, HScreenRectangle)

```
public void displayImage(HBackgroundImage image, HScreenRectangle r)
```

Display an image to cover a particular area of the screen. If the data for the image has not been loaded then this method will block while the data is loaded. It is platform dependent whether this image is scaled to fit or whether it is cropped (where too large) or repeated (where too small).

Parameters:

`image` - the image to display

`r` - the area of the screen to cover with the image

org.havi.ui
HSwitchable

Syntax
```
public interface HSwitchable extends HActionable
```

All Superinterfaces:
HActionable, HNavigable

All Known Implementing Classes:
HListElement, HToggleButton

Description
The HSwitchable interface is implemented for all user-interface components that can be toggled on or off (ie have four states, NORMAL_STATE , FOCUS_STATE , ACTION_STATE and NORMAL_ACTIONED_STATE)

The state transitions for an HSwitchable are as follows (initially, the HSwitchable is not actioned):

If the HSwitchable is in the FOCUS_STATE state then the action state transitions are as follows:

1. The HSwitchable is actioned when it receives an ACTION_PERFORMED java.awt.event.actionEvent.
2. The HSwitchable modifies its interaction state to ACTION_STATE .
3. The HSwitchable repaints itself to show any visual change in appearance, due to the associated HLook .
4. Any sound associated with the HSwitchable by the setActionSound(HSound) method shall be played. Note that it is not guaranteed that this sound will be played to completion, eg due to further user-interaction, etc.
5. Any ActionListeners are then called.
6. The HSwitchable remains in the ACTION_STATE state until it loses focus or it is actioned again.

If the HSwitchable is in the ACTION_STATE state then the action state transitions are as follows:

1. The HSwitchable is actioned when it receives an ACTION_PERFORMED java.awt.event.actionEvent.
2. The HSwitchable modifies its interaction state to FOCUS_STATE .
3. The HSwitchable repaints itself to show any visual change in appearance, due to the associated HLook .
4. Any sound associated with the HSwitchable by the setUnsetActionSound(HSound) method shall be played. Note that it is not guaranteed that this sound will be played to completion, eg due to further user-interaction, etc.
5. Any ActionListeners are then called.
6. The HSwitchable remains in the FOCUS_STATE state until it loses focus or it is actioned again.

If the HSwitchable is in the ACTION_STATE state then the focus state transitions are as follows:

1. The HSwitchable loses focus when it receives an java.awt.event.focusEvent of type FOCUS_LOST.
2. The HSwitchable modifies its interaction state to NORMAL_ACTIONED_STATE .
3. The HSwitchable repaints itself to show any visual change in appearance, due to the associated HLook .
4. Any sound associated with the HSwitchable by the setLoseFocusSound(HSound) method shall be played. Note that it is not guaranteed that this sound will be played to completion, eg due to further user-interaction, etc.
5. Any FocusListeners are then called.
6. The HSwitchable remains in the NORMAL_ACTIONED_STATE state until it gains focus or it is actioned

again.

If the HSwitchable is in the FOCUS_STATE state then the focus state transitions are as follows:

1. The HSwitchable loses focus when it receives an java.awt.event.focusEvent of type FOCUS_LOST.
2. The HSwitchable modifies its interaction state to NORMAL_STATE .
3. The HSwitchable repaints itself to show any visual change in appearance, due to the associated HLook .
4. Any sound associated with the HSwitchable by the setLoseFocusSound(HSound) method shall be played. Note that it is not guaranteed that this sound will be played to completion, eg due to further user-interaction, etc.
5. Any FocusListeners are then called.
6. The HSwitchable remains in the NORMAL_STATE state until it gains focus or it is actioned again.

If the HSwitchable is in the NORMAL_STATE state then the focus state transitions are as follows:

1. The HSwitchable gains focus when it receives an java.awt.event.focusEvent of type FOCUS_GAINED.
2. The HSwitchable modifies its interaction state to FOCUS_STATE .
3. The HSwitchable repaints itself to show any visual change in appearance, due to the associated HLook .
4. Any sound associated with the HSwitchable by the setGainFocusSound(HSound) method shall be played. Note that it is not guaranteed that this sound will be played to completion, eg due to further user-interaction, etc.
5. Any FocusListeners are then called.
6. The HSwitchable remains in the FOCUS_STATE state until it loses focus or it is actioned again.

If the HSwitchable is in the NORMAL_ACTIONED_STATE state then the focus state transitions are as follows:

1. The HSwitchable gains focus when it receives an java.awt.event.focusEvent of type FOCUS_GAINED.
2. The HSwitchable modifies its interaction state to ACTION_STATE .
3. The HSwitchable repaints itself to show any visual change in appearance, due to the associated HLook .
4. Any sound associated with the HSwitchable by the setGainFocusSound(HSound) method shall be played. Note that it is not guaranteed that this sound will be played to completion, eg due to further user-interaction, etc.
5. Any FocusListeners are then called.
6. The HSwitchable remains in the ACTION_STATE state until it loses focus or it is actioned again.

Subclasses of java.awt.Component which implement HSwitchable should by default enable both java.awt.event.FocusEvents and java.awt.event.ActionEvents. Whilst subclasses of java.awt.Component implementing HSwitchable may enable additional java.awt.AWTEvents, applications should assume that such classes only generate java.awt.event.FocusEvents and java.awt.event.ActionEvent, and should use the standard AWT mechanisms to enable additional events to be generated, if required.

In particular, the following classes implementing HSwitchable should all generate both java.awt.event.FocusEvent's and java.awt.event.HActionEvent's.

- HToggleButton
- HListElement

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

See Also:

HToggleButton

Methods

getSwitchableState()

```
public boolean getSwitchableState()
```

Returns the current switchable state of this HSwitchable .

Returns:
the current switchable state of this HSwitchable .

getUnsetActionSound()

```
public HSound getUnsetActionSound()
```

Get the sound to be played when the HSwitchable transitions from ACTION_STATE .

Returns:
the sound to be played when the HSwitchable transitions from ACTION_STATE .

setSwitchableState(boolean)

```
public void setSwitchableState(boolean state)
```

Sets the current state of the button. Note that ActionListeners are only called when an ACTION_PERFORMED event is received, or if they are called directly, e.g. via processActionEvent(), they are not called by setSwitchableState(boolean) .

setUnsetActionSound(HSound)

```
public void setUnsetActionSound(HSound sound)
```

Associate a sound to be played when the HSwitchable transitions from ACTION_STATE .

Parameters:
`sound` - a sound to be played when the HSwitchable transitions from ACTION_STATE . If sound content is already set, the original content is replaced. To remove the sound specify a null HSound .

org.havi.ui
HText

Syntax

```
public class HText extends HStaticText implements HNavigable
```

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--HComponent
              |
              +--HVisible
                    |
                    +--HStaticText
                          |
                          +--org.havi.ui.HText
```

Direct Known Subclasses:

HTextButton

All Implemented Interfaces:

HMatteLayer, HNavigable, HNoInputPreferred, HState, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

The HText class is used to display static, read-only, navigable, text. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	Width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Parameter	Description	Default value	Set method	Get method
height	Height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
text	The string to be used as the content for the NORMAL_STATE and FOCUS_STATE states of this component.	null	setTextContent(String, int)	getTextContent(int)
tlm	The text layout manager responsible for text formatting.	An HDefaultTextLayoutManager object.	setTextLayoutManager(HTextLayoutManager)	getTextLayoutManager()
font	The font for this component.	---	java.awt.Component#setFont	java.awt.Component#getFont
background	The background color for this component.	---	java.awt.Component#getBackground	java.awt.Component#setBackground
foreground	The foreground color for this component.	---	java.awt.Component#getForeground	java.awt.Component#setForeground

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The default "look" for this class.	A platform specific HTextLook	setDefaultLook(HTextLook)	getDefaultLook()
The "look" for this object.	The HGraphicLook returned from HText.getDefaultLook when this object was created.	setLook(HLook)	getLook()
The gain focus sound.	null	setGainFocusSound(HSound)	getGainFocusSound()
The lose focus sound.	null	setLoseFocusSound(HSound)	getLoseFocusSound()

Constructors

HText()

```
public HText()
```

Creates an HText object. See the class description for details of constructor parameters and default values.

HText(String)

```
public HText(java.lang.String text)
```

Creates an HText object. See the class description for details of constructor parameters and default values.

HText(String, Font, Color, Color, HTextLayoutManager)

```
public HText(java.lang.String text, java.awt.Font font, java.awt.Color foreground, java.
            awt.Color background, HTextLayoutManager tlm)
```

Creates an HText object. See the class description for details of constructor parameters and default values.

HText(String, int, int, int, int)

```
public HText(java.lang.String text, int x, int y, int width, int height)
```

Creates an HText object. See the class description for details of constructor parameters and default values.

HText(String, int, int, int, int, Font, Color, Color, HTextLayoutManager)

```
public HText(java.lang.String text, int x, int y, int width, int height, java.awt.
            Font font, java.awt.Color foreground, java.awt.Color background,
            HTextLayoutManager tlm)
```

Creates an HText object. See the class description for details of constructor parameters and default values.

Methods

getDefaultLook()

```
public static HTextLook getDefaultLook()
```

Returns the currently set default HLook for HText components.

Returns:
The HLook that is used by default when creating a new HText component.

getGainFocusSound()

```
public HSound getGainFocusSound()
```

Specified By:
getGainFocusSound() in interface HNavigable

See Also:
HNavigable

getLoseFocusSound()

```
public HSound getLoseFocusSound()
```

Specified By:
getLoseFocusSound() in interface HNavigable

See Also:
HNavigable

getMove(int)

```
public HNavigable getMove(int keyCode)
```

Specified By:
getMove(int) in interface HNavigable

See Also:
HNavigable

isFocusTraversable()

```
public boolean isFocusTraversable()
```

By default classes implementing HNavigable are focus-traversable.

Overrides:
isFocusTraversable() in class HVisible

Returns:
true

See Also:
`java.awt.Component.isFocusTraversable()`

isSelected()

```
public boolean isSelected()
```

Specified By:
isSelected() in interface HNavigable

See Also:
HNavigable

processEvent(AWTEvent)

```
protected void processEvent(java.awt.AWTEvent evt)
```

It is a valid implementation option for HText to override the protected processEvent method.

Overrides:
java.awt.Component.processEvent(java.awt.AWTEvent) in class java.awt.Component

See Also:
`java.awt.Component.processEvent(AWTEvent)`

processFocusEvent(FocusEvent)

```
protected void processFocusEvent(java.awt.event.FocusEvent evt)
```

It is a valid implementation option for HText to override the protected processFocusEvent method.

Overrides:
java.awt.Component.processFocusEvent(java.awt.event.FocusEvent) in class java.awt.Component

See Also:
`java.awt.Component.processFocusEvent(FocusEvent)`

requestFocus()

```
public void requestFocus()
```

Specified By:
requestFocus() in interface HNavigable

Overrides:
java.awt.Component.requestFocus() in class java.awt.Component

See Also:
`java.awt.Component.requestFocus()`

setDefaultLook(HTextLook)

```
public static void setDefaultLook(HTextLook look)
```

Sets the default HLook for all HText Components.

Parameters:
`look` - The HLook that will be used by default when creating a new HText component.

Throws:
HInvalidLookException - If the HLook is not an HTextLook .

setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
            HNavigable right)
```

Specified By:
setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) in interface HNavigable

See Also:
HNavigable

setGainFocusSound(HSound)

```
public void setGainFocusSound(HSound sound)
```

Specified By:
setGainFocusSound(HSound) in interface HNavigable

See Also:
HNavigable

setLoseFocusSound(HSound)

```
public void setLoseFocusSound(HSound sound)
```

Specified By:
setLoseFocusSound(HSound) in interface HNavigable

See Also:
HNavigable

setMove(int, HNavigable)

```
public void setMove(int keyCode, HNavigable target)
```

Specified By:
setMove(int, HNavigable) in interface HNavigable

See Also:
HNavigable

org.havi.ui
HTextButton

Syntax

```
public class HTextButton extends HText implements HActionable, HActionInputPreferred
```

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--HComponent
              |
              +--HVisible
                    |
                    +--HStaticText
                          |
                          +--HText
                                |
                                +--org.havi.ui.HTextButton
```

All Implemented Interfaces:

HActionable, HActionInputPreferred, HMatteLayer, HNavigable, HNoInputPreferred, HState, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

The HTextButton class is a conventional push- release textual button to be used for user actions. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	Width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
height	Height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Parameter	Description	Default value	Set method	Get method
text	The string to be used as the content for the NORMAL_STATE , FOCUS_STATE and ACTION_STATE states of this component.	null	setTextContent(String, int)	getTextContent(int)
tlm	The text layout manager responsible for text formatting.	An HDefaultTextLayoutManager object.	setTextLayoutManager(HTextLayoutManager)	getTextLayoutManager()
font	The font for this component.	---	java.awt.Component#setFont	java.awt.Component#getFont
background	The background color for this component.	---	java.awt.Component#getBackground	java.awt.Component#setBackground
foreground	The foreground color for this component.	---	java.awt.Component#getForeground	java.awt.Component#setForeground

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The default "look" for this class.	A platform specific HTextLook	setDefaultLook(HTextLook)	getDefaultLook()
The "look" for this object.	The HTextLook returned from HTextButton.getDefaultLook when this object was created.	setLook(HLook)	getLook()
The gain focus sound.	null	setGainFocusSound(HSound)	getGainFocusSound()
The lose focus sound.	null	setLoseFocusSound(HSound)	getLoseFocusSound()
The action sound.	null	setActionSound(HSound)	getActionSound()

Constructors

HTextButton()

```
public HTextButton()
```

Creates an HTextButton object. See the class description for details of constructor parameters and default values.

HTextButton(String)

```
public HTextButton(java.lang.String text)
```

Creates an HTextButton object. See the class description for details of constructor parameters and default values.

HTextButton(String, Font, Color, Color, HTextLayoutManager)

```
public HTextButton(java.lang.String text, java.awt.Font font, java.awt.Color foreground,
            java.awt.Color background, HTextLayoutManager tlm)
```

Creates an HTextButton object. See the class description for details of constructor parameters and default values.

HTextButton(String, int, int, int, int)

```
public HTextButton(java.lang.String text, int x, int y, int width, int height)
```

Creates an HTextButton object. See the class description for details of constructor parameters and default values.

HTextButton(String, int, int, int, int, Font, Color, Color, HTextLayoutManager)

```
public HTextButton(java.lang.String text, int x, int y, int width, int height, java.awt.
            Font font, java.awt.Color foreground, java.awt.Color background,
            HTextLayoutManager tlm)
```

Creates an HTextButton object. See the class description for details of constructor parameters and default values.

Methods

addActionListener(ActionListener)

```
public void addActionListener(java.awt.event.ActionListener l)
```

Specified By:
addActionListener(ActionListener) in interface HActionable

See Also:
HActionable

getActionCommand()

```
public java.lang.String getActionCommand()
```

Specified By:
getActionCommand() in interface HActionable

See Also:
HActionable

getActionSound()

```
public HSound getActionSound()
```

Specified By:
getActionSound() in interface HActionable

See Also:
HActionable

getDefaultLook()

```
public static HTextLook getDefaultLook()
```

Returns the currently set default HLook for HTextButton components.

Returns:
The HLook that is used by default when creating a new HTextButton .

getGainFocusSound()

```
public HSound getGainFocusSound()
```

Specified By:
getGainFocusSound() in interface HNavigable

Overrides:
getGainFocusSound() in class HText

See Also:
HNavigable

getLoseFocusSound()

```
public HSound getLoseFocusSound()
```

Specified By:
getLoseFocusSound() in interface HNavigable

Overrides:
getLoseFocusSound() in class HText

See Also:
HNavigable

getMove(int)

```
public HNavigable getMove(int keyCode)
```

Specified By:
getMove(int) in interface HNavigable

Overrides:
getMove(int) in class HText

See Also:
HNavigable

isFocusTraversable()

```
public boolean isFocusTraversable()
```

By default classes implementing HNavigable are focus-traversable.

Overrides:
isFocusTraversable() in class HText

Returns:
true

See Also:
`java.awt.Component.isFocusTraversable()`

isSelected()

```
public boolean isSelected()
```

Specified By:

isSelected() in interface HNavigable

Overrides:
isSelected() in class HText

See Also:
HNavigable

processActionEvent(ActionEvent)

```
protected void processActionEvent(java.awt.event.ActionEvent evt)
```

Processes an action event enabled for this object. Subclasses of classes implementing processActionEvent should call super.processActionEvent to ensure that the action event is handled appropriately.

Parameters:
`evt` - the event to be processed.

See Also:
HActionable

processEvent(AWTEvent)

```
protected void processEvent(java.awt.AWTEvent evt)
```

It is a valid implementation option for HTextButton to override the protected processEvent method.

Overrides:
processEvent(AWTEvent) in class HText

See Also:
`java.awt.Component.processEvent(AWTEvent)`

processFocusEvent(FocusEvent)

```
protected void processFocusEvent(java.awt.event.FocusEvent evt)
```

It is a valid implementation option for HTextButton to override the protected processFocusEvent method.

Overrides:
processFocusEvent(FocusEvent) in class HText

See Also:
`java.awt.Component.processFocusEvent(FocusEvent)`

removeActionListener(ActionListener)

```
public void removeActionListener(java.awt.event.ActionListener l)
```

Specified By:
removeActionListener(ActionListener) in interface HActionable

See Also:
HActionable

setActionCommand(String)

```
public void setActionCommand(java.lang.String command)
```

Specified By:
setActionCommand(String) in interface HActionable

See Also:
HActionable

setActionSound(HSound)

```
public void setActionSound(HSound sound)
```

Specified By:
setActionSound(HSound) in interface HActionable

See Also:
HActionable

setDefaultLook(HTextLook)

```
public static void setDefaultLook(HTextLook hlook)
```

Sets the default HLook for all HTextButton components.

Parameters:
`hlook` - The HLook that will be used by default when creating a new HTextButton .

Throws:
HInvalidLookException - If the HLook is not an HTextLook .

setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
            HNavigable right)
```

Specified By:
setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) in interface HNavigable

Overrides:
setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) in class HText

See Also:
HNavigable

setGainFocusSound(HSound)

```
public void setGainFocusSound(HSound sound)
```

Specified By:
setGainFocusSound(HSound) in interface HNavigable

Overrides:
setGainFocusSound(HSound) in class HText

See Also:
HNavigable

setLoseFocusSound(HSound)

```
public void setLoseFocusSound(HSound sound)
```

Specified By:
setLoseFocusSound(HSound) in interface HNavigable

Overrides:
setLoseFocusSound(HSound) in class HText

See Also:
HNavigable

setMove(int, HNavigable)

```
public void setMove(int keyCode, HNavigable target)
```

Specified By:

setMove(int, HNavigable) in interface HNavigable

Overrides:

setMove(int, HNavigable) in class HText

See Also:

HNavigable

org.havi.ui
HTextLayoutManager

Syntax
```
public interface HTextLayoutManager
```

All Known Implementing Classes:
DVBTextLayoutManager, HDefaultTextLayoutManager

Description
The HTextLayoutManager class manages the layout and rendering on-screen of a "marked-up" string.

Possible HTextLayoutManager behaviours could enable the following behaviours:

- Interpreting basic markup, such as changing color or font, and forced line breaks.
- Providing text alignment, as in the HDefaultTextLayoutManager .
- Providing text wrapping policies, such as word-wrap.
- Providing text orientations, such as right-to-left, or top-to-bottom rendering.
- Providing specialised support for missing characters, or fonts.
- Providing specific language support.
- Additional text styles, such as drop capitals or "shadow" characters.

See Also:
HDefaultTextLayoutManager

Methods

render(String, Graphics, HVisible)

```
public void render(java.lang.String markedUpString, java.awt.Graphics g, HVisible v)
```

Render the string, in the clipRect of the specified Graphics context. The HTextLayoutManager should not modify the clipRect of the Graphics object.

Parameters:

`markedUpString` - the string to render.

`g` - the graphics context, including its clipRect which encapsulates the x, y, width and height of the area within which rendering is permitted.

`v` - the HVisible into which to render.

org.havi.ui
HTextLook

Syntax
```
public class HTextLook implements HLook

java.lang.Object
  |
  +--org.havi.ui.HTextLook
```

All Implemented Interfaces:
java.lang.Cloneable, HLook

Description
The HTextLook class displays static read-only text on screen. This look will be provided by the platform and the exact way in which it is rendered will be platform dependant. However, the HTextLook associated with a given HVisible should use:

- java.awt.Component#getForeground() to determine the Color to render the rectangle.
- Border widths (as set by setHorizontalBorderSpacing / setVerticalBorderSpacing) of 2 pixels by default.
- HVisible.getTextLayoutManager() to determine the HTextLayoutManager to perform the rendering. If the returned object is null, then the text should not be rendered.

This is the default look that is used by HListElement , HStaticText and their subclasses. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

See Also:
HStaticText

Constructors

HTextLook()
```
public HTextLook()
```
Creates an HTextLook that will be used for rendering text.

Methods

getHorizontalBorderSpacing()

```
public int getHorizontalBorderSpacing()
```

Specified By:
getHorizontalBorderSpacing() in interface HLook

See Also:
HLook

getMaximumSize(HVisible)

```
public java.awt.Dimension getMaximumSize(HVisible visible)
```

Specified By:
getMaximumSize(HVisible) in interface HLook

See Also:
HLook

getMinimumSize(HVisible)

```
public java.awt.Dimension getMinimumSize(HVisible visible)
```

Specified By:
getMinimumSize(HVisible) in interface HLook

See Also:
HLook

getPreferredSize(HVisible)

```
public java.awt.Dimension getPreferredSize(HVisible visible)
```

Specified By:
getPreferredSize(HVisible) in interface HLook

See Also:
HLook

getVerticalBorderSpacing()

```
public int getVerticalBorderSpacing()
```

Specified By:
getVerticalBorderSpacing() in interface HLook

See Also:
HLook

setHorizontalBorderSpacing(int)

```
public void setHorizontalBorderSpacing(int width)
```

Specified By:
setHorizontalBorderSpacing(int) in interface HLook

See Also:
HLook

setVerticalBorderSpacing(int)

```
public void setVerticalBorderSpacing(int width)
```

Specified By:
setVerticalBorderSpacing(int) in interface HLook

See Also:
HLook

showLook(Graphics, HVisible, int)

```
public void showLook(java.awt.Graphics g, HVisible visible, int state)
```

Specified By:
showLook(Graphics, HVisible, int) in interface HLook

See Also:
HLook

org.havi.ui
HToggleButton

Syntax

```
public class HToggleButton extends HGraphicButton implements HSwitchable
```

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--HComponent
              |
              +--HVisible
                    |
                    +--HStaticIcon
                          |
                          +--HIcon
                                |
                                +--HGraphicButton
                                      |
                                      +--org.havi.ui.HToggleButton
```

All Implemented Interfaces:

HActionable, HActionInputPreferred, HMatteLayer, HNavigable, HNoInputPreferred, HState, HSwitchable, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

The HToggleButton class creates a "check box", or with the support of the HToggleGroup class, "radio buttons". The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default values exposed in the constructors

Parameter	Description	Default value	Set method	Get method
x	x-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
y	y-coordinate of top left hand corner of this component in pixels, relative to its parent container (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
width	Width of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds

Parameter	Description	Default value	Set method	Get method
height	Height of this component in pixels (subject to layout management).	---	java.awt.Component#setBounds	java.awt.Component#getBounds
image	The image to be used as the content for the NORMAL_STATE , FOCUS_STATE and ACTION_STATE states of this component.	null	setGraphicContent(Image, int)	getGraphicContent(int)
imageNormal	The image to be used as the content for the NORMAL_STATE state of this component.	null	setGraphicContent(Image, int)	getGraphicContent(int)
imagesFocused	The image to be used as the content for the FOCUS_STATE state of this component.	null	setGraphicContent(Image, int)	getGraphicContent(int)
imageActioned	The image to be used as the content for the ACTION_STATE state of this component.	null	setGraphicContent(Image, int)	getGraphicContent(int)
imageNormalActioned	The image to be used as the content for the NORMAL_ ACTIONED_STATE state of this component.	null	setGraphicContent(Image, int)	getGraphicContent(int)
state	The switchable state of this HToggleButton .	false	setSwitchableState(boolean)	getSwitchableState()
group	The HToggleGroup with which to associate this HToggleButton .	null	setToggleGroup(HToggleGroup)	getToggleGroup()

Default values not exposed in the constructors

Description	Default value	Set method	Get method
The default "look" for this class.	A platform specific HGraphicLook	setDefaultLook(HGraphicLook)	getDefaultLook()

Description	Default value	Set method	Get method
The "look" for this object.	The HGraphicLook returned from HToggleButton.getDefaultLook when this object was created.	setLook(HLook)	getLook()
The gain focus sound.	null	setGainFocusSound(HSound)	getGainFocusSound()
The lose focus sound.	null	setLoseFocusSound(HSound)	getLoseFocusSound()
The action sound.	null	setActionSound(HSound)	getActionSound()
The unset action sound.	null	setUnsetActionSound(HSound)	getUnsetActionSound()

Constructors

HToggleButton()

```
public HToggleButton()
```

Creates an HToggleButton object. See the class description for details of constructor parameters and default values.

HToggleButton(Image)

```
public HToggleButton(java.awt.Image image)
```

Creates an HToggleButton object. See the class description for details of constructor parameters and default values.

HToggleButton(Image, boolean, HToggleGroup)

```
public HToggleButton(java.awt.Image image, boolean state, HToggleGroup group)
```

Creates an HToggleButton object. See the class description for details of constructor parameters and default values.

HToggleButton(Image, Image, Image, Image, boolean)

```
public HToggleButton(java.awt.Image imageNormal, java.awt.Image imageFocused, java.awt.
            Image imageActioned, java.awt.Image imageNormalActioned, boolean state)
```

Creates an HToggleButton object. See the class description for details of constructor parameters and default values.

HToggleButton(Image, Image, Image, Image, boolean, HToggleGroup)

```
public HToggleButton(java.awt.Image imageNormal, java.awt.Image imageFocused, java.awt.
            Image imageActioned, java.awt.Image imageNormalActioned, boolean state,
            HToggleGroup group)
```

Creates an HToggleButton object. See the class description for details of constructor parameters and default values.

HToggleButton(Image, Image, Image, Image, int, int, int, int, boolean)

```
public HToggleButton(java.awt.Image imageNormal, java.awt.Image imageFocused, java.awt.
            Image imageActioned, java.awt.Image imageNormalActioned, int x, int y,
            int width, int height, boolean state)
```

Creates an HToggleButton object. See the class description for details of constructor parameters and default values.

HToggleButton(Image, Image, Image, Image, int, int, int, int, boolean, HToggleGroup)

```
public HToggleButton(java.awt.Image imageNormal, java.awt.Image imageFocused, java.awt.
            Image imageActioned, java.awt.Image imageNormalActioned, int x, int y,
            int width, int height, boolean state, HToggleGroup group)
```

Creates an HToggleButton object. See the class description for details of constructor parameters and default values.

HToggleButton(Image, int, int, int, int)

```
public HToggleButton(java.awt.Image image, int x, int y, int width, int height)
```

Creates an HToggleButton object. See the class description for details of constructor parameters and default values.

HToggleButton(Image, int, int, int, int, boolean)

```
public HToggleButton(java.awt.Image image, int x, int y, int width, int height,
            boolean state)
```

Creates an HToggleButton object. See the class description for details of constructor parameters and default values.

HToggleButton(Image, int, int, int, int, boolean, HToggleGroup)

```
public HToggleButton(java.awt.Image image, int x, int y, int width, int height,
            boolean state, HToggleGroup group)
```

Creates an HToggleButton object. See the class description for details of constructor parameters and default values.

Methods

addActionListener(ActionListener)

```
public void addActionListener(java.awt.event.ActionListener l)
```

Specified By:
addActionListener(ActionListener) in interface HActionable

Overrides:
addActionListener(ActionListener) in class HGraphicButton

See Also:
HActionable

getActionCommand()

```
public java.lang.String getActionCommand()
```

Specified By:
getActionCommand() in interface HActionable

Overrides:
getActionCommand() in class HGraphicButton

See Also:
HActionable

getActionSound()

```
public HSound getActionSound()
```

Specified By:
getActionSound() in interface HActionable

Overrides:
getActionSound() in class HGraphicButton

See Also:
HActionable

getDefaultLook()

```
public static HGraphicLook getDefaultLook()
```

Returns the currently set default HLook for HToggleButton components.

Returns:
The HLook that is used by default when creating a new HToggleButton component.

getGainFocusSound()

```
public HSound getGainFocusSound()
```

Specified By:
getGainFocusSound() in interface HNavigable

Overrides:
getGainFocusSound() in class HGraphicButton

See Also:
HNavigable

getLoseFocusSound()

```
public HSound getLoseFocusSound()
```

Specified By:
getLoseFocusSound() in interface HNavigable

Overrides:
getLoseFocusSound() in class HGraphicButton

See Also:
HNavigable

getMove(int)

```
public HNavigable getMove(int keyCode)
```

Specified By:
getMove(int) in interface HNavigable

Overrides:
getMove(int) in class HGraphicButton

See Also:
HNavigable

getSwitchableState()

```
public boolean getSwitchableState()
```

Specified By:
getSwitchableState() in interface HSwitchable

See Also:
HSwitchable

getToggleGroup()

```
public HToggleGroup getToggleGroup()
```

Gets the HToggleGroup the HToggleButton is associated with.

Returns:
The HToggleGroup the HToggleButton is associated with, or null if the HToggleButton is not associated with a HToggleGroup .

getUnsetActionSound()

```
public HSound getUnsetActionSound()
```

Specified By:
getUnsetActionSound() in interface HSwitchable

See Also:
HSwitchable

isFocusTraversable()

```
public boolean isFocusTraversable()
```

By default classes implementing HNavigable are focus-traversable.

Overrides:
isFocusTraversable() in class HGraphicButton

Returns:
true

See Also:
`java.awt.Component.isFocusTraversable()`

isSelected()

```
public boolean isSelected()
```

Specified By:
isSelected() in interface HNavigable

Overrides:
isSelected() in class HGraphicButton

See Also:
HNavigable

processActionEvent(ActionEvent)

```
protected void processActionEvent(java.awt.event.ActionEvent evt)
```

Processes an action event enabled for this object. Subclasses of classes implementing processActionEvent should call super.processActionEvent to ensure that the action event is handled appropriately.

Overrides:
processActionEvent(ActionEvent) in class HGraphicButton

Parameters:
`evt` - the event to be processed.

See Also:
HActionable

processEvent(AWTEvent)

```
protected void processEvent(java.awt.AWTEvent evt)
```

It is a valid implementation option for HToggleButton to override the protected processEvent method.

Overrides:
processEvent(AWTEvent) in class HGraphicButton

See Also:
`java.awt.Component.processEvent(AWTEvent)`

processFocusEvent(FocusEvent)

```
protected void processFocusEvent(java.awt.event.FocusEvent evt)
```

It is a valid implementation option for HToggleButton to override the protected processFocusEvent method.

Overrides:
processFocusEvent(FocusEvent) in class HGraphicButton

See Also:
`java.awt.Component.processFocusEvent(FocusEvent)`

removeActionListener(ActionListener)

```
public void removeActionListener(java.awt.event.ActionListener l)
```

Specified By:
removeActionListener(ActionListener) in interface HActionable

Overrides:
removeActionListener(ActionListener) in class HGraphicButton

See Also:
HActionable

removeToggleGroup()

```
public void removeToggleGroup()
```

Removes the button from the toggle group that it has been added to. This method does nothing if the button had not been previously added to an HToggleGroup .

setActionCommand(String)

```
public void setActionCommand(java.lang.String command)
```

Specified By:
setActionCommand(String) in interface HActionable

Overrides:
setActionCommand(String) in class HGraphicButton

See Also:
HActionable

setActionSound(HSound)

```
public void setActionSound(HSound sound)
```

Specified By:
setActionSound(HSound) in interface HActionable

Overrides:
setActionSound(HSound) in class HGraphicButton

See Also:
HActionable

setDefaultLook(HGraphicLook)

```
public static void setDefaultLook(HGraphicLook hlook)
```

Sets the default HLook for all HToggleButton components.

Parameters:
`hlook` - The HLook that will be used by default when creating a new HToggleButton component.

Throws:
HInvalidLookException - If the HLook is not an HGraphicLook .

setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable)

```
public void setFocusTraversal(HNavigable up, HNavigable down, HNavigable left,
            HNavigable right)
```

Specified By:
setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) in interface HNavigable

Overrides:
setFocusTraversal(HNavigable, HNavigable, HNavigable, HNavigable) in class HGraphicButton

See Also:
HNavigable

setGainFocusSound(HSound)

```
public void setGainFocusSound(HSound sound)
```

Specified By:
setGainFocusSound(HSound) in interface HNavigable

Overrides:
setGainFocusSound(HSound) in class HGraphicButton

See Also:
HNavigable

setLoseFocusSound(HSound)

```
public void setLoseFocusSound(HSound sound)
```

Specified By:

setLoseFocusSound(HSound) in interface HNavigable

Overrides:
setLoseFocusSound(HSound) in class HGraphicButton

See Also:
HNavigable

setMove(int, HNavigable)

```
public void setMove(int keyCode, HNavigable target)
```

Specified By:
setMove(int, HNavigable) in interface HNavigable

Overrides:
setMove(int, HNavigable) in class HGraphicButton

See Also:
HNavigable

setSwitchableState(boolean)

```
public void setSwitchableState(boolean state)
```

Specified By:
setSwitchableState(boolean) in interface HSwitchable

See Also:
HSwitchable

setToggleGroup(HToggleGroup)

```
public void setToggleGroup(HToggleGroup group)
```

Associates the HToggleButton with an HToggleGroup . If this HToggleButton is already in a different HToggleGroup , it is first taken out of that group.

Parameters:
`group` - The HToggleGroup the HToggleButton is to be associated with.

setUnsetActionSound(HSound)

```
public void setUnsetActionSound(HSound sound)
```

Specified By:
setUnsetActionSound(HSound) in interface HSwitchable

See Also:
HSwitchable

org.havi.ui
HToggleGroup

Syntax
```
public class HToggleGroup

java.lang.Object
  |
  +--org.havi.ui.HToggleGroup
```

Description
HToggleButton within the same HToggleGroup will behave so that a maximum of one HToggleButton has switchable state true, as returned by getSwitchableState(), so as to achieve a "radio button" effect. I. e. if a HToggleButton is acted upon to change its state to true, then if any other HToggleButton currently has state true, it will have its switchable state set to false. Similarly, if an HToggleButton is added which has switchable state true, then any current HToggleButton within the HToggleGroup with switchable state true, shall have its switchable state modified to false.

It is valid for all HToggleButton to have state false. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

None.

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Constructors

HToggleGroup()
```
public HToggleGroup()
```
Creates a new version instance of a HToggleGroup

Methods

getCurrent()
```
public HToggleButton getCurrent()
```
Returns the HToggleButton from this HToggleGroup which has state true, or null otherwise, for example, if there are no HToggleButton associated with this HToggleGroup , or if all HToggleButton within this HToggleGroup have state false.

setCurrent(HToggleButton)
```
public void setCurrent(HToggleButton selection)
```

If the specified HToggleButton is a member of this HToggleGroup , then it is selected, its state is set to true and consequently any other HToggleButton within the HToggleGroup will have their states set to false.

If the specified HToggleButton is not a member of this HToggleGroup , then no actions are performed.

Parameters:

`selection` - the HToggleButton to be set as the currently selected item within the HToggleGroup .

org.havi.ui
HUIException

Syntax
```
public class HUIException extends java.lang.Exception

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.havi.ui.HUIException
```

Direct Known Subclasses:
HInvalidLookException, HMatteException

All Implemented Interfaces:
java.io.Serializable

Description
HUIException is a generic exception that indicates that the desired user-interface mechanism cannot be performed for some reason. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Constructors

HUIException()
```
public HUIException()
```

org.havi.ui
HValue

Syntax
```
public interface HValue extends HNavigable
```

All Superinterfaces:
HNavigable

All Known Implementing Classes:
HListGroup, HRangeValue, HSinglelineEntry

Description
This interface is implemented for all widget types that represent some form of value (eg a range control or text edit). A HValue widget can register a HValueChangeListener that will send events when the value is modified by the user (e.g. the range has changed or the text has been changed).

The state of widgets implementing the HValue interface which are being modified shall be FOCUS_STATE .

Subclasses of java.awt.Component which implement HValue should by default enable java.awt.event.FocusEvents, java.awt.event.KeyEvents and HValueChangeEvent . Whilst subclasses of java.awt.Component implementing HValue may enable additional java.awt.AWTEvents, applications should assume that such classes only generate java.awt.event.FocusEvents, java.awt.event.KeyEvents and HValueChangeEvent , and should use the standard AWT mechanisms to enable additional events to be generated, if required.

In particular, the following classes implementing HValue should all generate both java.awt.event.FocusEvent's, java.awt.event.KeyEvents and HValueChangeEvent .

- HSinglelineEntry
- HMultilineEntry
- HRange

Classes implementing HValue should also implement an additional protected method, as follows:

```
protected void processActionEvent(org.havi.ui.event.HValueChangeEvent evt)
```
in order to process HValueChangeEvent .

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

See Also:
HAdjustmentInputPreferred

Methods

addChangeListener(HValueChangeListener)

```
public void addChangeListener(HValueChangeListener l)
```

Adds the specified HValueChangeListener to receive HValueChangeEvent from this object.

Parameters:
`l` - the HValueChangeListener to be notified.

getChangeSound()

```
public HSound getChangeSound()
```

Get the sound to be played on reception of an HValueChangeEvent .

Returns:
The sound played when the components value changes

removeChangeListener(HValueChangeListener)

```
public void removeChangeListener(HValueChangeListener l)
```

Removes the specified HValueChangeListener so that it no longer receives HValueChangeEvent from this object. If the specified listener is not registered, the method has no effect.

Parameters:
`l` - the HValueChangeListener to be removed from notification.

setChangeSound(HSound)

```
public void setChangeSound(HSound sound)
```

Associate a sound to be played on reception of an HValueChangeEvent .

Parameters:
`sound` - the sound to be played, when the component value is modified. If sound content is already set, the original content is replaced. To remove the sound specify a null HSound .

org.havi.ui
HVersion

Syntax
```
public interface HVersion
```

Description
The HVersion interface defines some versioning constants that are accessible by using the java.lang. System method getProperty, with the appropriate property name.

Note that it is a valid implementation to return empty strings for the implementation, vendor and name strings.

Fields

HAVI_IMPLEMENTATION_NAME

```
public static final java.lang.String HAVI_IMPLEMENTATION_NAME
```

A string constant describing the HAVi implementation name, as returned via java.lang.System.getProperty(havi.implementation.name).

HAVI_IMPLEMENTATION_VENDOR

```
public static final java.lang.String HAVI_IMPLEMENTATION_VENDOR
```

A string constant describing the HAVi implementation vendor, as returned via java.lang.System.getProperty(havi.implementation.vendor).

HAVI_IMPLEMENTATION_VERSION

```
public static final java.lang.String HAVI_IMPLEMENTATION_VERSION
```

A string constant describing the HAVi implementation version, as returned via java.lang.System.getProperty(havi.implementation.version).

HAVI_SPECIFICATION_NAME

```
public static final java.lang.String HAVI_SPECIFICATION_NAME
```

A string constant describing the HAVi specification name, as returned via java.lang.System.getProperty(havi.specification.name).

HAVI_SPECIFICATION_VENDOR

```
public static final java.lang.String HAVI_SPECIFICATION_VENDOR
```

A string constant describing the HAVi specification vendor, as returned via java.lang.System.getProperty(havi.specification.vendor).

HAVI_SPECIFICATION_VERSION

```
public static final java.lang.String HAVI_SPECIFICATION_VERSION
```

A string constant describing the HAVi specification version, as returned via java.lang.System.getProperty(havi.specification.version).

org.havi.ui
HVideoComponent

Syntax
```
public class HVideoComponent extends HComponent

java.lang.Object
  |
  +--java.awt.Component
        |
        +--HComponent
              |
              +--org.havi.ui.HVideoComponent
```

All Implemented Interfaces:
HMatteLayer, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description
HVideoComponent is an opaque class encapsulating the presentation of a video source *within* an application, i.e. contained within a conventional AWT hierarchy.

A HVideoComponent obeys all conventional java.awt.Component semantics, including being clipped by its parent container, etc. A HVideoComponent also obeys all HComponent semantics including Z-ordering, etc. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Constructors

HVideoComponent()
```
protected HVideoComponent()
```
It is not intended that applications should directly construct HVideoComponent objects. Creates an HVideoComponent object. See the class description for details of constructor parameters and default values.

Methods

addOnScreenLocationModifiedListener(HScreenLocationModifiedListener)

```
public void addOnScreenLocationModifiedListener(HScreenLocationModifiedListener slml)
```

Register a listener to determine if the Component's on-screen location is modified - irrespective of its relative location to its parent Container.

getVideoDevice()

```
public HVideoDevice getVideoDevice()
```

Returns the HVideoDevice that this HVideoComponent is associated with.

Returns:

the HVideoDevice that this HVideoComponent is associated with, or null if this cannot be determined.

removeOnScreenLocationModifiedListener(HScreenLocationModifiedListener)

```
public void removeOnScreenLocationModifiedListener(HScreenLocationModifiedListener slml)
```

Remove a listener that determines if the Component's on-screen location is modified - irrespective of its relative location to its parent Container. If the specified listener is not registered, the method has no effect.

org.havi.ui
HVideoConfigTemplate

Syntax

```
public class HVideoConfigTemplate extends HScreenConfigTemplate

java.lang.Object
  |
  +--HScreenConfigTemplate
        |
        +--org.havi.ui.HVideoConfigTemplate
```

Description

The HVideoConfigTemplate class is used to obtain a valid HVideoConfiguration . An application instantiates one of these objects and then sets all non-default attributes as desired. The object is then passed to the getBestConfiguration(HVideoConfigTemplate) method found in the HVideoDevice class. A valid HVideoConfiguration is returned that meets if possible which exceeds what was requested in the HVideoConfigTemplate .

This class this may be subclassed to support define additional properties of video configurations which may be requested by applications. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Fields

GRAPHICS_MIXING

```
public static final int GRAPHICS_MIXING
```

A value for use in the preference field of the setPreference(int, int) method in the HVideoConfigTemplate that indicates that the device configuration supports the display of graphics in addition to video streams. This display includes both configurations where the video pixels and graphics pixels are fully aligned (same size) as well as configurations where they are displayed together but where a more complex relationship exists between the two pixel coordinate spaces. The graphics device for mixing is specified as an HGraphicsConfiguration .

Constructors

HVideoConfigTemplate()

```
public HVideoConfigTemplate()
```

Methods

getBestConfiguration(HVideoConfiguration[])

```
public HVideoConfiguration getBestConfiguration(HVideoConfiguration[] hvca)
```

Returns the "best" configuration possible that passes the criteria defined in this HVideoConfigTemplate .

Best in this sense means satisfying the preferences in the config template as follows based on the priority (as supplied to HScreenConfigTemplate.setPreference())

1. satisfying all the preferences in that config template whose was REQUIRED
2. excluding configurations with priorites which were REQUIRED_NOT
3. satisfying as many as possible of the preferences whose priority was PREFERRED .
4. Satisfying as few as possible of the preferences whose priority was PREFERRED_NOT .

Parameters:
`hvca` - the array of HVideoConfiguration objects to choose from, represented as an array of HVideoConfiguration objects.

Returns:
an HVideoConfiguration object that is the best configuration possible, as a HVideoConfiguration object.

isVideoConfigSupported(HVideoConfiguration)

```
public boolean isVideoConfigSupported(HVideoConfiguration hvc)
```

Returns a boolean indicating whether or not the specified HVideoConfiguration can be used to create a drawing surface that supports the indicated features.

Parameters:
`hvc` - - the HVideoConfiguration represented as a HVideoConfiguration object to test

Returns:
true if this HVideoConfiguration object can be used to create surfaces that support the indicated features; false if the HVideoConfiguration can not be used to create a drawing surface usable by this application.

See Also:
HVideoConfigTemplate

org.havi.ui
HVideoConfiguration

Syntax
```
public class HVideoConfiguration extends HScreenConfiguration

java.lang.Object
  |
  +--HScreenConfiguration
        |
        +--org.havi.ui.HVideoConfiguration
```

Description
The HVideoConfiguration class describes the characteristics (settings) of an HVideoDevice . There can be many HVideoConfiguration objects associated with a single HVideoDevice . The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

See Also:
HVideoDevice

Constructors

HVideoConfiguration()
```
protected HVideoConfiguration()
```
It is not intended that applications should directly construct HVideoConfiguration objects. Creates an HVideoConfiguration object. See the class description for details of constructor parameters and default values.

Methods

getConfigTemplate()
```
public HVideoConfigTemplate getConfigTemplate()
```
Returns an HVideoConfigTemplate object that describes and uniquely identifies this HVideoConfiguration .

Hence, the following sequence should return the original HVideoConfiguration .

```
HVideoDevice.getBestMatch(HVideoConfiguration .getConfigTemplate())
```

Properties that are implemented in the HVideoConfiguration will return REQUIRED priority, features that are not implemented in the HBackgroundConfiguration will return REQUIRED_NOT priority.

Returns:

an HVideoConfigTemplate object which both describes and uniquely identifies this HVideoConfiguration .

getDevice()

```
public HVideoDevice getDevice()
```

Returns the HVideoDevice associated with this HVideoConfiguration .

Returns:

the HVideoDevice object that is associated with this HVideoConfiguration ,

org.havi.ui
HVideoDevice

Syntax
```
public class HVideoDevice extends HScreenDevice

java.lang.Object
  |
  +--HScreenDevice
        |
        +--org.havi.ui.HVideoDevice
```

All Implemented Interfaces:
org.davic.resources.ResourceProxy, org.davic.resources.ResourceServer

Description
The HVideoDevice class describes the logical video devices which can contribute to the appearance of a particular screen. Each HVideoDevice has one or more HVideoConfiguration objects associated with it. These objects specify the different configurations (settings) in which the HVideoDevice can be used. This class represents the presentation only of video and does not provide for the selection of which video is to be presented. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

See Also:
HVideoConfiguration, HScreen

Constructors

HVideoDevice()
```
protected HVideoDevice()
```
It is not intended that applications should directly construct HVideoDevice objects. Creates an HVideoDevice object. See the class description for details of constructor parameters and default values.

Methods

getBestConfiguration(HVideoConfigTemplate)

```
public HVideoConfiguration getBestConfiguration(HVideoConfigTemplate hvct)
```

Returns the "best" configuration possible that passes the criteria defined in the HVideoConfigTemplate .

Best in this sense means satisfying the preferences in the config template as follows based on the priority (as supplied to HScreenConfigTemplate.setPreference())

1. satisfying all the preferences in that config template whose was REQUIRED
2. excluding configurations with priorites which were REQUIRED_NOT
3. satisfying as many as possible of the preferences whose priority was PREFERRED .
4. Satisfying as few as possible of the preferences whose priority was PREFERRED_NOT .

Parameters:
`hvct` - an HVideoConfigTemplate object used to obtain a valid HVideoConfiguration .

Returns:
an HVideoConfiguration that passes the criteria defined in the specified HVideoConfigTemplate ,

getBestConfiguration(HVideoConfigTemplate[])

```
public HVideoConfiguration getBestConfiguration(HVideoConfigTemplate[] hvcta)
```

Returns the "best" configuration possible that passes the criteria defined in one of the HVideoConfigTemplate objects within the specified array. The HVideoTemplate objects should be considered for matching in priority order from 0 to (hvcta.length - 1).

Best in this sense means satisfying the preferences in the config template as follows based on the priority (as supplied to HScreenConfigTemplate.setPreference())

1. satisfying all the preferences in that config template whose was REQUIRED
2. excluding configurations with priorites which were REQUIRED_NOT
3. satisfying as many as possible of the preferences whose priority was PREFERRED .
4. Satisfying as few as possible of the preferences whose priority was PREFERRED_NOT .

Parameters:
`hvcta` - the HVideoConfigTemplate array used to obtain a valid HVideoConfiguration .

Returns:
an HVideoConfiguration that passes the criteria defined in one of the HVideoConfigTemplate objects within the specified array

getConfigurations()

```
public HVideoConfiguration[] getConfigurations()
```

Returns all of the HVideoConfiguration objects associated with this HVideoDevice .

Returns:
an array of HVideoConfiguration objects

See Also:
HVideoConfiguration

getCurrentConfiguration()

```
public HVideoConfiguration getCurrentConfiguration()
```

Returns the current HVideoConfiguration for this HVideoDevice .

Returns:

the current HVideoConfiguration for this HVideoDevice .

See Also:
HVideoConfiguration

getDefaultConfiguration()

```
public HVideoConfiguration getDefaultConfiguration()
```

Returns the default HVideoConfiguration associated with this HVideoDevice . This (single) default configuration should correspond to some well-behaved settings for the device, such as, a minimal configuration or factory preset settings.

Returns:
the default HVideoConfiguration of this HVideoDevice .

See Also:
HVideoConfiguration

getVideoController()

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

Obtain a reference to the object which controls the presentation of the video. Null is returned if no video is being presented. In systems based on JMF, this would be the javax.media.Player instance which owns the resource.

Returns:
the object which controls the presentation of the video

Throws:
`SecurityException` - if the application does not have sufficient rights to get the VideoPlayer object.

HPermissionDeniedException - (HPermissionDeniedException) if the application does not currently have the right to get the VideoPlayer object.

getVideoSource()

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

Obtain a reference to the source of the video being presented by this device at this moment. The precise class to be be returned must be specified outside the HAVi user- interface specification. Null is returned if no video is being presented.

Returns:
a reference to the source of the video

Throws:
`SecurityException` - if the application does not have sufficient rights to get the VideoSource object.

HPermissionDeniedException - (HPermissionDeniedException) if the application does not currently have the right to get the VideoSource object.

setVideoConfiguration(HVideoConfiguration)

```
public boolean setVideoConfiguration(HVideoConfiguration hvc)
```

Set the video configuration for the device.

Parameters:
`hvc` - the HVideoConfiguration to which this device should be set.

Returns:

A boolean indicating whether the configuration could be applied successfully. If the configuration could not be applied successfully, the configuration after this method may not match the configuration of the device prior to this method being called --- applications should take steps to determine whether a partial change of settings has been made.

Throws:

`SecurityException` - if the application does not have sufficient rights to set the configuration for this device.

HPermissionDeniedException - (HPermissionDeniedException) if the application does not currently have the right to set the configuration for this device.

HConfigurationException - (HConfigurationException) if the specified configuration is not valid for this device.

org.havi.ui
HVisible

Syntax

```
public class HVisible extends HComponent implements HState
```

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--HComponent
              |
              +--org.havi.ui.HVisible
```

Direct Known Subclasses:

HListElement, HSinglelineEntry, HStaticAnimation, HStaticIcon, HStaticRange, HStaticText

All Implemented Interfaces:

HMatteLayer, HState, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

Description

The HVisible class is the base class for all non- interactive component, ie it only has a single (unfocused) state, denoted as NORMAL_STATE .

Whilst implementations of HVisible may enable certain java.awt.AWTEvents, applications should assume that an HVisible class does not generate any java.awt.AWTEvents and should use the standard AWT mechanisms to enable such events to be generated, if required. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Constructors

HVisible()

```
public HVisible()
```

Creates a HVisible component with no HLook . The size and location of the component will be controlled by the Layout manager associated with the Container into which this component is placed.

HVisible(HLook)

```
public HVisible(HLook hlook)
```

Creates a HVisible component with the given HLook .

Parameters:

`hlook` - The HLook associated with the HVisible component.

HVisible(HLook, int, int, int, int)

```
public HVisible(HLook hlook, int x, int y, int width, int height)
```

Creates a HVisible component with the given HLook and the specified location and size.

Parameters:

`hlook` - The HLook associated with the HVisible component.

`x` - the x-coordinate of the HVisible component within its Container.

`y` - the y-coordinate of the HVisible component within its Container.

`width` - the width of the HVisible component in pixels.

`height` - the height of the HVisible component in pixels.

Methods

getAnimateContent(int)

```
public java.awt.Image[] getAnimateContent(int state)
```

Gets the animate content for this component.

Parameters:

`The` - state for which content is to be retrieved.

Returns:

The animate content associated with the specified state. If no animate content has been set for the specified state, then null is returned.

getContent(int)

```
public java.lang.Object getContent(int state)
```

Gets the content for this component.

Parameters:

`state` - The state for which content is to be retrieved.

Returns:

The content associated with the specified state. If no content has been set for the specified state, then null is returned.

getGraphicContent(int)

```
public java.awt.Image getGraphicContent(int state)
```

Gets the graphic content for this component.

Parameters:

`state` - The state for which content is to be retrieved.

Returns:

The graphical content associated with the specified state. If no graphical content has been set for the specified state, then null is returned.

getInteractionState()

```
public int getInteractionState()
```

Return the interaction state the widget is currently in.

Returns:
the interaction state the widget is currently in.

See Also:
NORMAL_STATE, FOCUS_STATE, ACTION_STATE, NORMAL_ACTIONED_STATE

getLook()

```
public HLook getLook()
```

Gets the HLook for this component.

Returns:
the HLook that is being used by this component --- if no HLook has been set, then returns null.

getMaximumSize()

```
public java.awt.Dimension getMaximumSize()
```

Gets the maximum size of the HVisible . The getMaximumSize method of the HLook that is associated with this HVisible will be called to calculate the dimensions.

Overrides:
java.awt.Component.getMaximumSize() in class java.awt.Component

Returns:
A dimension object indicating this HVisible 's maximum size --- if no HLook has been associated with the HVisible , then the current HVisible dimension determined via getSize() will be returned.

See Also:
getMaximumSize(HVisible)

getMinimumSize()

```
public java.awt.Dimension getMinimumSize()
```

Gets the minimum size of the HVisible . The getMinimumSize method of the HLook that is associated with this HVisible will be called to calculate the dimensions.

Overrides:
java.awt.Component.getMinimumSize() in class java.awt.Component

Returns:
A dimension object indicating this HVisible 's minimum size --- if no HLook has been associated with the HVisible , then the current HVisible dimension determined via getSize() will be returned.

See Also:
getMinimumSize(HVisible)

getPreferredSize()

```
public java.awt.Dimension getPreferredSize()
```

Gets the preferred size of the HVisible . The getPreferredSize method of the HLook that is associated with this HVisible will be called to calculate the dimensions.

Overrides:

java.awt.Component.getPreferredSize() in class java.awt.Component

Returns:
A dimension object indicating this HVisible 's preferred size --- if no HLook has been associated with the HVisible , then the current HVisible dimension determined via getSize() will be returned.

See Also:
getPreferredSize(HVisible)

getTextContent(int)

```
public java.lang.String getTextContent(int state)
```

Gets the text content for this component.

Parameters:
`state` - The state for which content is to be retrieved.

Returns:
The text content associated with the specified state. If no text content has been set for the specified state, then null is returned.

getTextLayoutManager()

```
public HTextLayoutManager getTextLayoutManager()
```

Gets the text layout manager that is being used to layout this text.

Returns:
The HTextLayoutManager that is being used by this component.

isFocusTraversable()

```
public boolean isFocusTraversable()
```

By default HVisible is not focus-traversable.

Overrides:
java.awt.Component.isFocusTraversable() in class java.awt.Component

Returns:
false

See Also:
`java.awt.Component.isFocusTraversable()`

paint(Graphics)

```
public void paint(java.awt.Graphics g)
```

Draws the current state of the component, by calling the HLook showLook(Graphics, HVisible, int) method. The java.awt.Component paint method shall directly call the showLook(Graphics, HVisible, int) method of the associated HLook . If no HLook is associated with the widget, then the paint method should do nothing. Similarly, if the HVisible is created with a null HLook , then the paint method does nothing. These mechanisms may be used for Components that wish to extend HVisible , and override the paint method, without supporting the HLook interface.

Overrides:
java.awt.Component.paint(java.awt.Graphics) in class java.awt.Component

Parameters:
`g` - the specified context to use for updating.

setAnimateContent(Image[], int)

```
public void setAnimateContent(java.awt.Image[] imageArray, int state)
```

Sets an array of graphical content (primarily used for animation), per state. Different (single arrays of) content can be associated with the different states of a widget.

Note that the content is not copied, merely its object reference.

If the HVisible has an associated HLook , then it should invoke its showLook(Graphics, HVisible, int) method. It is implementation specific whether setting multiple content collapses multiple invocations of showLook(Graphics, HVisible, int) .

Parameters:

`imageArray` - An array of images that make up the animation. If the array is null, then any currently assigned content shall be removed for the specified state.

`state` - The state of the widget for which this content should be displayed.

setContent(Object, int)

```
public void setContent(java.lang.Object object, int state)
```

Sets a single piece of content for this component, per state. Different (single pieces of) content can be associated with the different states of a widget.

Note that the content is not copied, merely its object reference.

If the HVisible has an associated HLook , then it should invoke its showLook(Graphics, HVisible, int) method. It is implementation specific whether setting multiple content collapses multiple invocations of showLook(Graphics, HVisible, int) .

Parameters:

`object` - The content. If the content is null, then any currently assigned content shall be removed for the specified state.

`state` - The state of the widget for which this content should be displayed.

setGraphicContent(Image, int)

```
public void setGraphicContent(java.awt.Image image, int state)
```

Sets a single piece of graphical content for this component, per state. Different (single pieces of) content can be associated with the different states of a widget.

Note that the content is not copied, merely its object reference.

If the HVisible has an associated HLook , then it should invoke its showLook(Graphics, HVisible, int) method. It is implementation specific whether setting multiple content collapses multiple invocations of showLook(Graphics, HVisible, int) .

Parameters:

`image` - The content. If the content is null, then any currently assigned content shall be removed for the specified state.

`state` - The state of the widget for which this content should be displayed.

setInteractionState(int)

```
protected void setInteractionState(int state)
```

Set the interaction state for this widget.

Parameters:

`state` - the interaction state for this widget.

See Also:

NORMAL_STATE, FOCUS_STATE, ACTION_STATE, NORMAL_ACTIONED_STATE

setLook(HLook)

```
public void setLook(HLook hlook)
```

Sets the HLook for this component.

Parameters:

`hlook` - The HLook that is to be used for this component.

Throws:

HInvalidLookException - If the Look is not compatable with this type of component, for example a graphic look being set on a text component.

setTextContent(String, int)

```
public void setTextContent(java.lang.String string, int state)
```

Sets a single piece of text content for this component, per state. Different (single pieces of) content can be associated with the different states of a widget.

Note that the content is not copied, merely its object reference.

If the HVisible has an associated HLook , then it should invoke its showLook(Graphics, HVisible, int) method. It is implementation specific whether setting multiple content collapses multiple invocations of showLook(Graphics, HVisible, int) .

Parameters:

`string` - The content. If the content is null, then any currently assigned content shall be removed for the specified state.

`state` - The state of the widget for which this content should be displayed.

setTextLayoutManager(HTextLayoutManager)

```
public void setTextLayoutManager(HTextLayoutManager manager)
```

Sets the text layout manager that should be used to layout the text for this component.

Parameters:

`manager` - the HTextLayoutManager to be used by this component.

update(Graphics)

```
public void update(java.awt.Graphics g)
```

The update() method in HVisible overrides that in Component and does not clear the background of the widget, it simply modifies the current Color of the Graphics object to match that of the widgets background Color, and calls the paint() method.

Overrides:

java.awt.Component.update(java.awt.Graphics) in class java.awt.Component

Package org.havi.ui.event

Class Summary

Interfaces

HBackgroundImageListener	The listener interface for receiving events related to HBackgroundImage objects.
HScreenConfigurationListener	This listener is used to monitor when an HScreenDevice configuration is modified.
HScreenLocationModifiedListener	This listener is used to monitor when a component, such as an HVideoComponent on-screen location is modified.
HValueChangeListener	The HValueChangeListener interface enables the reception of HValueChangeEvent , as generated by objects implementing HValue.

Classes

HBackgroundImageEvent	This event informs an application that the data for an HBackgroundImage has been loaded.
HEventRepresentation	This class is able to describe the representation of an event generator as a string, color or symbol (such as a triangle, '>', for 'play').
HKeyCapabilities	This class is used to describe the (basic) keyboard capabilities of the platform.
HMouseCapabilities	This class is used to describe the (basic) mouse capabilities of the platform.
HRcCapabilities	This class is used to describe the (basic) remote control capabilities of the platform.
HRcEvent	The remote control event class.
HScreenConfigurationEvent	This event is sent to all registered HScreenConfigurationListener when an HScreenDevice modifies its configuration.
HScreenDeviceReleasedEvent	This event informs an application that a device for this HScreen has been released by an application or other entity in the system.
HScreenDeviceReservedEvent	This event informs that a device on this HScreen has been reserved by an application or other entity in the system.
HScreenLocationModifiedEvent	This event is generated by the system when a component is moved on-screen, rather than within a container.
HUIEvent	The HUIEvent class encapsulates the events to which all HAVi widgets respond in addition to conventional Java AWT mechanisms.
HValueChangeEvent	An HValueChangeEvent is sent from a widget implementing the HValue interface to any registered HValueChangeListener .

org.havi.ui.event

HBackgroundImageEvent

Syntax

```
public class HBackgroundImageEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.havi.ui.event.HBackgroundImageEvent
```

All Implemented Interfaces:

java.io.Serializable

Description

This event informs an application that the data for an HBackgroundImage has been loaded. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

None.

Default values not exposed in the constructors

None.

Constructors

HBackgroundImageEvent(Object)

```
public HBackgroundImageEvent(java.lang.Object im)
```

Constructor for the event.

Parameters:

`im` - the HBackgroundImage which has been loaded.

Methods

---

getSource()

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

Returns the HBackgroundImage for which the data has been loaded.

Overrides:

java.util.EventObject.getSource() in class java.util.EventObject

Returns:

the object which has been loaded.

org.havi.ui.event

HBackgroundImageListener

Syntax

```
public interface HBackgroundImageListener extends java.util.EventListener
```

All Superinterfaces:

java.util.EventListener

Description

The listener interface for receiving events related to HBackgroundImage objects.

Methods

---

imageLoaded(HBackgroundImageEvent)

```
public void imageLoaded(HBackgroundImageEvent e)
```

Invoked when the data for an HBackgroundImage has been loaded.

Parameters:

`e` - the event which happened

org.havi.ui.event
HEventRepresentation

Syntax

```
public class HEventRepresentation

java.lang.Object
  |
  +--org.havi.ui.event.HEventRepresentation
```

Description

This class is able to describe the representation of an event generator as a string, color or symbol (such as a triangle, '>', for 'play'). This allows an application to describe a button on an input device correctly for a given platform.

The particular text, color, or symbol can be determined by calling getString, getColor or getSymbol respectively. All available events should have a text representation from getString.

The six colored key events (VK_COLORED_KEY_1 -- VK_COLORED_KEY_6) , if implemented, must also be represented by a color - the getColor method returns a java.awt.Color object.

Key events may also be represented as a symbol - if the platform does not support a symbolic representation for a given event, then the application is responsible for rendering the symbol itself. The rendering of keys with a commonly known representation should follow the guidelines given here, as defined in the following table.

VK_REWIND

VK_STOP

VK_PAUSE

VK_PLAY

VK_FAST_FWD

VK_GO_TO_END

VK_PREV_TRACK

VK_NEXT_TRACK

VK_RECORD

VK_EJECT_TOGGLE

VK_VOLUME_UP

VK_VOLUME_DOWN

VK_UP

VK_DOWN

VK_LEFT

VK_RIGHT

VK_POWER

Event	Implied symbol

VK_GO_TO_START	Two equilateral triangles, pointing at a line to the left
Two equilateral triangles, pointing to the left	
A square	
Two vertical lines, side by side	
One equilateral triangle, pointing to the right	
Two equilateral triangles, pointing to the right	
Two equilateral triangles, pointing to a line at the right	
One equilateral triangle, pointing to a line at the left	
One equilateral triangle, pointing to a line at the right	
A circle, normally red	
A line under a wide triangle which points up	
A ramp, increasing to the right, near a plus sign	
A ramp, increasing to the right, near a minus sign	
An arrow pointing up	
An arrow pointing down	
An arrow pointing to the left	
An arrow pointing to the right	
A circle, broken at the top, with a vertical line in the break	

The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

None.

Default values not exposed in the constructors

None.

Fields

ER_TYPE_COLOR

```
public static final int ER_TYPE_COLOR
```

The event representation type for the current event is supported as a color.

See Also:
ER_TYPE_NOT_SUPPORTED

ER_TYPE_NOT_SUPPORTED

```
public static final int ER_TYPE_NOT_SUPPORTED
```

The event representation type for the current event is not supported.

The four ER_TYPE integers describe if an input mechanism is not supported, or is described by a string, color or symbol.

See Also:
ER_TYPE_STRING, ER_TYPE_COLOR, ER_TYPE_SYMBOL

ER_TYPE_STRING

```
public static final int ER_TYPE_STRING
```

The event representation type for the current event is supported as a string.

See Also:
ER_TYPE_NOT_SUPPORTED

ER_TYPE_SYMBOL

```
public static final int ER_TYPE_SYMBOL
```

The event representation type for the current event is supported as a symbol.

See Also:
ER_TYPE_NOT_SUPPORTED

Constructors

HEventRepresentation()

```
protected HEventRepresentation()
```

Constructs an HEventRepresentation object.

Methods

getColor()

```
public java.awt.Color getColor()
```

This returns the color representation (generally used for colored soft keys) of the current event code.

Returns:
The color representation of the current event code, or null if not available.

---

getString()

```
public java.lang.String getString()
```

Returns the text representation of the current event code.

Returns:
The text representation of the current event code, or null if not available.

---

getSymbol()

```
public java.awt.Image getSymbol()
```

This returns an image-based representation (generally used for symbolic keys) of the current event code.

Note that it is platform specific whether this method will return a valid Image, in particular it is a valid implementation option to always return null. Note that for non-null Images, the size and other Image characteristics are dependent on particular manufacturer implementation.

Returns:
The symbolic representation of the current event code, or null if not available.

---

getType()

```
public int getType()
```

This returns the type of representation(s) available for the current event code.

If the event code can be represented in multiple ways, then the returned type will be the sum of the supported types, e.g. an event generated by a key with a particular font representation of an "A" in yellow might return ER_TYPE_STRING + ER_TYPE_COLOR + ER_TYPE_SYMBOL. Where the string representation is "A", the color representation is "yellow" and the symbol representation might be a likeness of the "A glyph" from a particular font.

---

isSupported()

```
public boolean isSupported()
```

This method returns true if the current event is supported by the platform.

---

setColor(Color)

```
protected void setColor(java.awt.Color aColor)
```

Sets the Color representation for this HEventRepresentation . Any previous value is overwritten.

---

setString(String)

```
protected void setString(java.lang.String aText)
```

Sets the string representation for this HEventRepresentation . Any previous value is overwritten.

setSymbol(Image)

```
protected void setSymbol(java.awt.Image Symbol)
```

Sets the symbolic representation for this HEventRepresentation . Any previous value is overwritten.

setType(int)

```
protected void setType(int aType)
```

Sets the event id for this HEventRepresentation . Any previous value is overwritten.

org.havi.ui.event
HKeyCapabilities

Syntax
```
public class HKeyCapabilities

java.lang.Object
  |
  +--org.havi.ui.event.HKeyCapabilities
```

Direct Known Subclasses:
HRcCapabilities

Description
This class is used to describe the (basic) keyboard capabilities of the platform.

This class is not intended to be constructed by applications. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

None.

Default values not exposed in the constructors

None.

Constructors

HKeyCapabilities()
```
protected HKeyCapabilities()
```
It is not intended that applications should directly construct HKeyCapabilities objects. Creates an HKeyCapabilities object. See the class description for details of constructor parameters and default values.

Methods

getInputDeviceSupported()
```
public static boolean getInputDeviceSupported()
```
Returns true if external keyboard functionality exists in the system, false otherwise.

Returns:
true if external keyboard functionality exists in the system, false otherwise.

isSupported(int)
```
public static boolean isSupported(int keycode)
```

Queries whether the system can ever generate an event of the given type.

Parameters:

`keycode` - the virtual keycode to query e.g. VK_SPACE

Returns:

true if events with the given key code can (ever) be generated on this system.

org.havi.ui.event
HMouseCapabilities

Syntax
```
public class HMouseCapabilities

java.lang.Object
  |
  +--org.havi.ui.event.HMouseCapabilities
```

Description
This class is used to describe the (basic) mouse capabilities of the platform.

This class is not intended to be constructed by applications. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

None.

Default values not exposed in the constructors

None.

Constructors

HMouseCapabilities()

```
protected HMouseCapabilities()
```

It is not intended that applications should directly construct HMouseCapabilities objects. Creates an HMouseCapabilities object. See the class description for details of constructor parameters and default values.

Methods

getInputDeviceSupported()

```
public static boolean getInputDeviceSupported()
```

This returns true if a mouse exists in the system.

org.havi.ui.event
HRcCapabilities

Syntax
```
public class HRcCapabilities extends HKeyCapabilities

java.lang.Object
  |
  +--HKeyCapabilities
        |
        +--org.havi.ui.event.HRcCapabilities
```

Description
This class is used to describe the (basic) remote control capabilities of the platform.

This class is not intended to be constructed by applications. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

None.

Default values not exposed in the constructors

None.

Constructors

HRcCapabilities()
```
protected HRcCapabilities()
```
It is not intended that applications should directly construct HRcCapabilities objects. Creates an HRcCapabilities object. See the class description for details of constructor parameters and default values.

Methods

getInputDeviceSupported()
```
public static boolean getInputDeviceSupported()
```
This returns true if a remote control exists in the system.

getRepresentation(int)
```
public static HEventRepresentation getRepresentation(int aCode)
```
Get the HEventRepresentation object for a specified key event id.

Parameters:

`aCode` - the key event id for which the HEventRepresentation should be returned. A null object shall be returned if the specified key event id does not have a valid remote control representation.

org.havi.ui.event
HRcEvent

Syntax

```
public class HRcEvent extends java.awt.event.KeyEvent
```

```
java.lang.Object
  |
  +--java.util.EventObject
        |
        +--java.awt.AWTEvent
              |
              +--java.awt.event.ComponentEvent
                    |
                    +--java.awt.event.InputEvent
                          |
                          +--java.awt.event.KeyEvent
                                |
                                +--org.havi.ui.event.HRcEvent
```

All Implemented Interfaces:

java.io.Serializable

Description

The remote control event class.

The presence or absence of these keys and their desired representation is provided by the HRcCapabilities class.

Note that it is an implementation option if remote control key events are repeated.

Note that the reception of these events by a java.awt.Component is dependent on it having java.awt.event.KeyEvent's enabled.

Note that it is an implementation constraint that the HRcEvent event range should not intersect with the Java AWT key event ranges, or the HUIEvent event range. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Fields

RC_FIRST

```
public static final int RC_FIRST
```

Marks the first integer id for the range of remote control event ids.

RC_LAST

```
public static final int RC_LAST
```

Marks the last integer id for the range of remote control event ids.

VK_BALANCE_LEFT

```
public static final int VK_BALANCE_LEFT
```

The 'balance left' action id - moves the audio balance to the left.

VK_BALANCE_RIGHT

```
public static final int VK_BALANCE_RIGHT
```

The 'balance right' action id - moves the audio balance to the right.

VK_BASS_BOOST_DOWN

```
public static final int VK_BASS_BOOST_DOWN
```

The 'bass boost down' action id - decreases the audio amplifier bass boost.

VK_BASS_BOOST_UP

```
public static final int VK_BASS_BOOST_UP
```

The 'bass boost up' action id - increases the audio amplifier bass boost.

VK_CHANNEL_DOWN

```
public static final int VK_CHANNEL_DOWN
```

The 'channel down' action id.

VK_CHANNEL_UP

```
public static final int VK_CHANNEL_UP
```

The 'channel up' action id.

VK_CLEAR_FAVORITE_0

```
public static final int VK_CLEAR_FAVORITE_0
```

The 'clear favorite 0' action id.

VK_CLEAR_FAVORITE_1

```
public static final int VK_CLEAR_FAVORITE_1
```

The 'clear favorite 1' action id.

VK_CLEAR_FAVORITE_2

```
public static final int VK_CLEAR_FAVORITE_2
```

The 'clear favorite 2' action id.

VK_CLEAR_FAVORITE_3

```
public static final int VK_CLEAR_FAVORITE_3
```

The 'clear favorite 3' action id.

VK_COLORED_KEY_0

```
public static final int VK_COLORED_KEY_0
```

Colored key 0 action id.

Up to six colored soft keys can be included on a remote control. These are optional, and must be identified with a color. If implemented, these keys are to be oriented from left to right, or from top to bottom in ascending order.

The application can determine how many colored keys are implemented, and what colors are to be used, so that the application can match the controls, by using the getRepresentation method in the RcCapabilities class.

See Also:
VK_COLORED_KEY_1, VK_COLORED_KEY_2, VK_COLORED_KEY_3, VK_COLORED_KEY_4, VK_COLORED_KEY_5

VK_COLORED_KEY_1

```
public static final int VK_COLORED_KEY_1
```

Colored key 1 action id.

See Also:
VK_COLORED_KEY_0

VK_COLORED_KEY_2

```
public static final int VK_COLORED_KEY_2
```

Colored key 2 action id.

See Also:
VK_COLORED_KEY_0

VK_COLORED_KEY_3

```
public static final int VK_COLORED_KEY_3
```

Colored key 3 action id.

See Also:
VK_COLORED_KEY_0

VK_COLORED_KEY_4

```
public static final int VK_COLORED_KEY_4
```

Colored key 4 action id.

See Also:
VK_COLORED_KEY_0

VK_COLORED_KEY_5

```
public static final int VK_COLORED_KEY_5
```

Colored key 5 action id.

See Also:
VK_COLORED_KEY_0

VK_DIMMER

```
public static final int VK_DIMMER
```

The 'device dimmer' action id adjusts illumination of the device.

This may be a toggle between two states, or a sequence through multiple states.

VK_DISPLAY_SWAP

```
public static final int VK_DISPLAY_SWAP
```

The 'display swap' action id - swaps displayed video sources.

VK_EJECT_TOGGLE

```
public static final int VK_EJECT_TOGGLE
```

The 'eject / insert media' action id.

VK_FADER_FRONT

```
public static final int VK_FADER_FRONT
```

The 'fader front' action id - moves the audio fader to the front.

VK_FADER_REAR

```
public static final int VK_FADER_REAR
```

The 'fader rear' action id - moves the audio fader to the rear.

VK_FAST_FWD

```
public static final int VK_FAST_FWD
```

The 'fast forward (media)' action id.

VK_GO_TO_END

```
public static final int VK_GO_TO_END
```

The '(send media) to end position' action id.

VK_GO_TO_START

```
public static final int VK_GO_TO_START
```

The 'go (send media) to start position' action id.

VK_GUIDE

```
public static final int VK_GUIDE
```

The 'guide' action id - indicates a user request for a programme guide (toggle).

VK_HELP

```
public static final int VK_HELP
```

The 'help' action id - indicates that the user has requested assistance (toggle).

VK_INFO

```
public static final int VK_INFO
```

The 'info' action id - indicates that the user has requested additional information (toggle).

VK_MUTE

```
public static final int VK_MUTE
```

The 'mute' action id - mute audio output

VK_PAUSE

```
public static final int VK_PAUSE
```

The 'pause (media)' action id.

VK_PINP_TOGGLE

```
public static final int VK_PINP_TOGGLE
```

The 'picture in picture toggle' action id - turns picture in picture mode on or off.

VK_PLAY

```
public static final int VK_PLAY
```

The 'play (media)' action id.

VK_PLAY_SPEED_DOWN

```
public static final int VK_PLAY_SPEED_DOWN
```

The 'decrease (media) play speed' action id.

VK_PLAY_SPEED_RESET

```
public static final int VK_PLAY_SPEED_RESET
```

The 'set (media) play speed to normal' action id.

VK_PLAY_SPEED_UP

```
public static final int VK_PLAY_SPEED_UP
```

The 'increase (media) play speed' action id.

VK_POWER

```
public static final int VK_POWER
```

The 'device power' action id turns on or off the delegated device.

VK_RANDOM_TOGGLE

```
public static final int VK_RANDOM_TOGGLE
```

The 'toggle random (media) play' action id.

VK_RECALL_FAVORITE_0

```
public static final int VK_RECALL_FAVORITE_0
```

The 'recall favorite 0' action id.

VK_RECALL_FAVORITE_1

```
public static final int VK_RECALL_FAVORITE_1
```

The 'recall favorite 1' action id.

VK_RECALL_FAVORITE_2

```
public static final int VK_RECALL_FAVORITE_2
```

The 'recall favorite 2' action id.

VK_RECALL_FAVORITE_3

```
public static final int VK_RECALL_FAVORITE_3
```

The 'recall favorite 3' action id.

VK_RECORD

```
public static final int VK_RECORD
```

The 'record (to media)' action id.

VK_RECORD_SPEED_NEXT

```
public static final int VK_RECORD_SPEED_NEXT
```

The 'select next (media) record speed' action id.

VK_REWIND

```
public static final int VK_REWIND
```

The 'rewind (media)' action id.

VK_SCAN_CHANNELS_TOGGLE

```
public static final int VK_SCAN_CHANNELS_TOGGLE
```

The 'scan channels toggle' action id - turns channel scanning on or off.

VK_SCREEN_MODE_NEXT

```
public static final int VK_SCREEN_MODE_NEXT
```

The 'screen mode next' action id - advances the display screen mode.

VK_SPLIT_SCREEN_TOGGLE

```
public static final int VK_SPLIT_SCREEN_TOGGLE
```

The 'split screen toggle' action id - turns split screen on or off.

VK_STOP

```
public static final int VK_STOP
```

The 'stop (media)' action id.

VK_STORE_FAVORITE_0

```
public static final int VK_STORE_FAVORITE_0
```

The 'store current setting as favorite 0' action id.

VK_STORE_FAVORITE_1

```
public static final int VK_STORE_FAVORITE_1
```

The 'store current setting as favorite 1' action id.

VK_STORE_FAVORITE_2

```
public static final int VK_STORE_FAVORITE_2
```

The 'store current setting as favorite 2' action id.

VK_STORE_FAVORITE_3

```
public static final int VK_STORE_FAVORITE_3
```

The 'store current setting as favorite 3' action id.

VK_SUBTITLE

```
public static final int VK_SUBTITLE
```

The 'subtitle' action id - indicates a user request for subtitling (toggle).

VK_SURROUND_MODE_NEXT

```
public static final int VK_SURROUND_MODE_NEXT
```

The 'surround mode next' action id - advances audio amplifier surround mode.

VK_TELETEXT

```
public static final int VK_TELETEXT
```

The 'teletext' action id - indicates a user request for a teletext service (toggle).

VK_TRACK_NEXT

```
public static final int VK_TRACK_NEXT
```

The '(send media) to next track' action id.

VK_TRACK_PREV

```
public static final int VK_TRACK_PREV
```

The '(send media) to previous track' action id.

VK_VIDEO_MODE_NEXT

```
public static final int VK_VIDEO_MODE_NEXT
```

The 'video mode next' action id - advances the display video mode.

VK_VOLUME_DOWN

```
public static final int VK_VOLUME_DOWN
```

The 'volume down' action id - decreases audio amplifier volume.

VK_VOLUME_UP

```
public static final int VK_VOLUME_UP
```

The 'volume up' action id - increases audio amplifier volume.

VK_WINK

```
public static final int VK_WINK
```

The 'device wink' action id is used to indicated that the device should identify itself in some manner, for example, audibly or visually.

Constructors

HRcEvent(Component, int, long, int, int)

```
public HRcEvent(java.awt.Component source, int id, long when, int modifiers, int keyCode)
```

Deprecated.
See explanation in java.awt.event.KeyEvent.

Constructs an HRcEvent object with the specified source component, type, modifiers and key.

Parameters:

`source` - the object where the event originated.

`id` - the identifier.

`when` - the time stamp for this event.

`modifiers` - indication of any modification keys that are active for this event.

`keyCode` - the code of the key associated with this event.

HRcEvent(Component, int, long, int, int, char)

```
public HRcEvent(java.awt.Component source, int id, long when, int modifiers, int keyCode,
            char keyChar)
```

Constructs an HRcEvent object with the specified source component, type, modifiers and key.

Parameters:

`source` - the object where the event originated.

`id` - the identifier.

`when` - the time stamp for this event.

`modifiers` - indication of any modification keys that are active for this event.

`keyCode` - the code of the key associated with this event.

`keyChar` - the character representation of the key associated with this event.

org.havi.ui.event
HScreenConfigurationEvent

Syntax
```
public class HScreenConfigurationEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.havi.ui.event.HScreenConfigurationEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
This event is sent to all registered HScreenConfigurationListener when an HScreenDevice modifies its configuration. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

None.

Default values not exposed in the constructors

None.

See Also:
HScreenConfigurationListener, HScreenDevice

Constructors

HScreenConfigurationEvent(Object)

```
public HScreenConfigurationEvent(java.lang.Object source)
```

Construct an HScreenConfigurationEvent

Parameters:
`source` - the HScreenDevice whose configuration changed

org.havi.ui.event
HScreenConfigurationListener

Syntax
```
public interface HScreenConfigurationListener extends java.util.EventListener
```

All Superinterfaces:
java.util.EventListener

Description
This listener is used to monitor when an HScreenDevice configuration is modified. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

See Also:
HScreenConfigurationEvent, HScreenDevice

Methods

---

report(HScreenConfigurationEvent)

```
public void report(HScreenConfigurationEvent gce)
```

org.havi.ui.event
HScreenDeviceReleasedEvent

Syntax
```
public class HScreenDeviceReleasedEvent extends org.davic.resources.ResourceStatusEvent

java.lang.Object
  |
  +--org.davic.resources.ResourceStatusEvent
        |
        +--org.havi.ui.event.HScreenDeviceReleasedEvent
```

Description
This event informs an application that a device for this HScreen has been released by an application or other entity in the system. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

None.

Default values not exposed in the constructors

None.

Constructors

HScreenDeviceReleasedEvent(Object)

```
public HScreenDeviceReleasedEvent(java.lang.Object bg)
```

Constructor for the event

Parameters:

`bg` - the HScreenDevice which has been released

Methods

---

getSource()

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

Returns the device that has been released

Overrides:

org.davic.resources.ResourceStatusEvent.getSource() in class org.davic.resources.ResourceStatusEvent

Returns:

the HScreenDevice object representing the device that has been released

org.havi.ui.event
HScreenDeviceReservedEvent

Syntax
```
public class HScreenDeviceReservedEvent extends org.davic.resources.ResourceStatusEvent

java.lang.Object
  |
  +--org.davic.resources.ResourceStatusEvent
        |
        +--org.havi.ui.event.HScreenDeviceReservedEvent
```

Description
This event informs that a device on this HScreen has been reserved by an application or other entity in the system. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

None.

Default values not exposed in the constructors

None.

Constructors

HScreenDeviceReservedEvent(Object)

```
public HScreenDeviceReservedEvent(java.lang.Object bg)
```

Constructor for the event

Parameters:

`bg` - the HScreenDevice representing the device which has been reserved

Methods

---

getSource()

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

Returns the device that has been reserved

Overrides:

org.davic.resources.ResourceStatusEvent.getSource() in class org.davic.resources.ResourceStatusEvent

Returns:

an HScreenDevice representing the device that has been reserved

org.havi.ui.event
HScreenLocationModifiedEvent

Syntax
```
public class HScreenLocationModifiedEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.havi.ui.event.HScreenLocationModifiedEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
This event is generated by the system when a component is moved on-screen, rather than within a container. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

None.

Default values not exposed in the constructors

None.

Constructors

HScreenLocationModifiedEvent(Object)

```
public HScreenLocationModifiedEvent(java.lang.Object source)
```

Constructor for the event

Parameters:

`source` - the Component whose on-screen location has been modified.

Methods

---

getSource()

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

Overrides:

java.util.EventObject.getSource() in class java.util.EventObject

org.havi.ui.event
HScreenLocationModifiedListener

Syntax
```
public interface HScreenLocationModifiedListener extends java.util.EventListener
```

All Superinterfaces:
java.util.EventListener

Description
This listener is used to monitor when a component, such as an HVideoComponent on-screen location is modified. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

See Also:
HScreenLocationModifiedEvent, HVideoComponent

Methods

---

report(HScreenLocationModifiedEvent)

```
public void report(HScreenLocationModifiedEvent gce)
```

org.havi.ui.event
HUIEvent

Syntax

```
public class HUIEvent extends java.awt.event.KeyEvent
```

```
java.lang.Object
  |
  +--java.util.EventObject
        |
        +--java.awt.AWTEvent
              |
              +--java.awt.event.ComponentEvent
                    |
                    +--java.awt.event.InputEvent
                          |
                          +--java.awt.event.KeyEvent
                                |
                                +--org.havi.ui.event.HUIEvent
```

All Implemented Interfaces:

java.io.Serializable

Description

The HUIEvent class encapsulates the events to which all HAVi widgets respond in addition to conventional Java AWT mechanisms. This mechanism is required for systems which lack a pointing device, and strongly recommended for systems which do.

Note that it is an implementation constraint that the HUIEvent event range should not intersect with the Java AWT key event ranges, or the HRcEvent event range. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Default values not exposed in the constructors

Parameter	Description	Default value	Set method	Get method

Fields

UI_FIRST

```
public static final int UI_FIRST
```

Marks the first integer id for the range of UI event ids. The range of UI event ids should not clash with other subclasses of KeyEvent on a per-platform basis.

UI_LAST

```
public static final int UI_LAST
```

Marks the last integer id for the range of UI event ids.

VK_ACTION

```
public static final int VK_ACTION
```

This value indicates that the widget should become actioned.

VK_ADJUST_LESS

```
public static final int VK_ADJUST_LESS
```

This value indicates that the widget should decrease its value.

VK_ADJUST_MORE

```
public static final int VK_ADJUST_MORE
```

This value indicates that the widget should increase its value.

VK_END_CHANGE

```
public static final int VK_END_CHANGE
```

This value indicates that the widget should switch out of an (internal) changeable / editing mode (ditto)

VK_NEXT_CHAR

```
public static final int VK_NEXT_CHAR
```

This value indicates that the widget should move the caret forward one character (ditto)

VK_NEXT_LINE

```
public static final int VK_NEXT_LINE
```

This value indicates that the widget should move the caret down one line (ditto)

VK_PREV_CHAR

```
public static final int VK_PREV_CHAR
```

This value indicates that the widget should move the caret back one character (ditto)

VK_PREV_LINE

```
public static final int VK_PREV_LINE
```

This value indicates that the widget should move the caret up one line (ditto)

VK_START_CHANGE

```
public static final int VK_START_CHANGE
```

This value indicates that the widget should switch to an (internal) changeable / editing mode (for HSinglelineEntry etc.)

Constructors

HUIEvent(Component, int, long, int, int)

```
public HUIEvent(java.awt.Component source, int id, long when, int modifiers, int keyCode)
```

Deprecated.
See explanation in java.awt.event.KeyEvent.

Constructs an HUIEvent object with the specified source component, type, modifiers and key.

Parameters:

`source` - the object where the event originated.

`id` - the identifier.

`when` - the time stamp for this event.

`modifiers` - indication of any modification keys that are active for this event.

`keyCode` - the code of the key associated with this event.

HUIEvent(Component, int, long, int, int, char)

```
public HUIEvent(java.awt.Component source, int id, long when, int modifiers, int keyCode,
            char keyChar)
```

Constructs an HUIEvent object with the specified source component, type, modifiers and key.

Parameters:

`source` - the object where the event originated.

`id` - the identifier.

`when` - the time stamp for this event.

`modifiers` - indication of any modification keys that are active for this event.

`keyCode` - the code of the key associated with this event.

`keyChar` - the character representation of the key associated with this event.

org.havi.ui.event
HValueChangeEvent

Syntax
```
public class HValueChangeEvent extends java.awt.AWTEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--java.awt.AWTEvent
              |
              +--org.havi.ui.event.HValueChangeEvent
```

All Implemented Interfaces:
java.io.Serializable

Description
An HValueChangeEvent is sent from a widget implementing the HValue interface to any registered HValueChangeListener . The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Default parameter values exposed in the constructors

None.

Default values not exposed in the constructors

None.

Fields

VALUE_CHANGE
```
public static final int VALUE_CHANGE
```
The value change event, which is sent whenever the value of an HValue widget is modified.

VALUE_ENDCHANGE
```
public static final int VALUE_ENDCHANGE
```
The final value change event, which is sent when the value of an HValue widget has been finally set.

VALUE_FIRST
```
public static final int VALUE_FIRST
```
The first integer id for the range of value id's supported by the HValueChangeEvent class.

VALUE_LAST
```
public static final int VALUE_LAST
```
The last integer id for the range of value id's supported by the HValueChangeEvent class.

VALUE_STARTCHANGE

```
public static final int VALUE_STARTCHANGE
```

The start value change event, which is sent when the value of an HValue widget can be initially modified by the user. That is, it signifies that the user has requested to modify the value of the widget. This event is generated by an HSinglelineEntry widget when entering its editable mode.

Constructors

HValueChangeEvent(HValue, int)

```
public HValueChangeEvent(HValue source, int id)
```

Constructs a HValueChangeEvent .

Parameters:

`source` - The HValue widget whose value has been modified.

`id` - The event id of the HValueChangeEvent generated by the HValue widget. This is the value that will be returned by the object's getID method.

org.havi.ui.event
HValueChangeListener

Syntax
```
public interface HValueChangeListener extends java.util.EventListener
```

All Superinterfaces:
java.util.EventListener

Description
The HValueChangeListener interface enables the reception of HValueChangeEvent , as generated by objects implementing HValue. The parameters to the constructors are as follows, in cases where parameters are not used, then the constructor should use the default values.

Methods

---

valueChanged(HValueChangeEvent)

```
public void valueChanged(HValueChangeEvent e)
```

Called when the value of the component has started to be changed, is changed, or has finished changing (eg text added or a range value changed).

Parameters:
`e` - is the HValueChangeEvent generated by the an object implementing HValue.

Annex W (normative): Application life cycle example

W.1 DVB-J Application lifecycle implementation example

```
package com.random.myxlet;

/**
 * This pseudocode illustrates the kind of actions that an Xlet should
 * take in response to Xlet state change callbacks.  It is primarily meant
 * to illustrate what is to be done in the paused state.  An example of
 * the kind of application that might use this kind of resource-management
 * policy is a slide-show application that is used to view images.  In this
 * example, the application caches only the current image when in the
 * paused state, but caches more images when in the active state.
 */

import javax.tv.Xlet;
import javax.tv.XletContext;

public class Sample implements Xlet {

   private boolean started = false;       // Latch, set true when first started
   private boolean active = false;        // Condition variable synchronized on this
   private boolean destroyed = false;     // Latch, set true when destroyed
   private XletContext context;

   public void initXlet(XletContext ctx) {
      context = ctx;
   }

   public synchronized void destroyXlet(boolean unconditional) {
      active = false;
   }

   private void manageCache() {
      for (;;) {
      synchronized(this) {
         if (destroyed) {
         ... free any resources we hold ...;
         cancel any image loading currently in progress;
         return;                          // Terminates thread
      }
      if (!active) {
         free all cached images, except the one currently
            being viewed;
      } else {                            // active
         if (cached images remain to be loaded
               && an image load is not in process)
         {
         Start an image load.  Provide the image loader
         with a listener that will, when the image loading
         is complete, notify us and do this.notifyAll().
         }
      }
      wait();
      }
```

```
    }
    }
}
```

A simple example of Xlet lifecycle is a stock ticker application that uses a back channel to retrieve stock quotes, which it displays on the viewer's television.

a) The application manager retrieves the Xlet's code.

b) The application manager creates an instance of the XletContext Object and initializes it for the new Xlet.

c) The application manager initializes the Xlet by calling its `initXlet()` method and passing it the context object.

d) The Xlet uses the context object to initialize itself and enters the `Paused` state.

e) The application manager calls the Xlet's `startXlet()` method. The application manager assumes that the Xlet is performing its service.

f) Upon receiving this signal, the Xlet creates a new thread that opens the back channel to retrieve the stock quotes. The Xlet is now in the `Active` state.

g) The Xlet begins to show the stock quotes.

h) Due to circumstances beyond the control of the Xlet, it is no longer able to retrieve updated stock quotes.

i) The Xlet decides to continue displaying the most recent quotes it has. Note that the Xlet is still in the `Active` state.

j) After a time, the Xlet is still unable to open the back channel. It decides that the quotes it is displaying are too old to present and that it can no longer perform its service. It chooses to take itself out of the `Active` state. It calls the `paused()` method on XletContext to signal this change to the application manager.

k) Finally, the Xlet decides it no longer has any chance of performing its service, so it decides it should be terminated. It calls the `destroyed()` method on the XletContext to signal application manager that it has entered the `Destroyed` state. The Xlet does some final clean up.

l) The application manager prepares the Xlet for garbage collection.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z

History

Document history		
V1.1.1	July 2000	Publication