ETSI TS 102 727 V1.1.1 (2010-01)

Technical Specification

Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.2.2

Reference
DTS/JTC-DVB-264

Keywords
broadcast, DVB

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 ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

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

Foreword

This Technical Specification (TS) has been produced by 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

The Digital Video Broadcasting Project (DVB) is an industry-led consortium of broadcasters, manufacturers, network operators, software developers, regulatory bodies, content owners and others committed to designing global standards for the delivery of digital television and data services. DVB fosters market driven solutions that meet the needs and economic circumstances of broadcast industry stakeholders and consumers. DVB standards cover all aspects of digital television from transmission through interfacing, conditional access and interactivity for digital video, audio and data. The consortium came together in 1993 to provide global standardisation, interoperability and future proof specifications.

1 Scope

The present document defines the DVB solution for Multimedia Home Platforms (MHPs) that was developed to fulfil the related DVB commercial requirements MHP045. As with previous versions of the MHP specification it relies on the use of appropriate DVB specifications for digital video broadcast and associated interactive services TR 101 200 [i.4]. The MHP is applicable to all DVB defined transmission media and networks such as satellite, cable, terrestrial, microwave and TCP/IP.

1.1 Document structure

Clauses 1 to 14 specify the applicable technologies and technical definitions in a generic way. Clause 15 provides detailed profile definitions for the following initial profiles:

- Enhanced Broadcasting 3;
- Interactive Broadcasting 3;
- Internet Access 3;
- IPTV 3;

which can be extended with future additional profile definitions.

Clause 16 provides a registry of constants and clause 17 describes requirements for internet access clients.

The present document is firstly intended for implementers of MHP terminals 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 the present document regarding conformance.

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

One of the primary goals of the present document is to minimize the number of divergences between MHP and GEM terminal specifications, wherever practical. Divergence is defined in clause 3.1. Where divergences are inescapable, the present document serves as a place to document and control the permitted divergences, so that they will be predictable to terminal manufacturers, broadcasters, and application authors.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific.

- For a specific reference, subsequent revisions do not apply.
- Non-specific reference may be made only to a complete document or a part thereof and only in the following cases:
  - if it is accepted that it will be possible to use all future changes of the referenced document for the purposes of the referring document;
  - for informative references.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

Some known errata in these references are identified in annex A "External references; errata, clarifications and exemptions". These errata take precedence over the published reference.

2.1 Normative references

The following referenced documents are indispensable for the application of the present document. For dated references, only the edition cited applies. For non-specific references, the latest edition of the referenced document (including any amendments) applies.

[1] ETSI TS 102 728 (V1.1.1): "Digital Video Broadcasting (DVB); Globally Executable MHP (GEM) Specification 1.2.2 (including IPTV)".

[2] CORBA/IIOP: "The Common Object Request Broker: Architecture and Specification", Object Management Group.

NOTE: Available at http://www.omg.org/cgi-bin/doc?formal/04-03-12.pdf.

[3] DAVIC 1.4.1p9, 1999: "DAVIC 1.4.1 Specification Part 9: Information Representation", DAVIC.

NOTE: Available at http://portal.etsi.org/docbox/Reference/DAVIC/.

[4] ETSI EN 300 468 (V1.5.1): "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems".

[5] ETSI EN 301 192 (V1.3.1): "Digital Video Broadcasting (DVB); DVB specification for data broadcasting".

[6] ETSI EN 301 193 (V1.1.1): "Digital Video Broadcasting (DVB); Interaction channel through the Digital Enhanced Cordless Telecommunications (DECT)".

[7] ETSI EN 301 195 (V1.1.1): "Digital Video Broadcasting (DVB); Interaction channel through the Global System for Mobile communications (GSM)".

[8] ETSI EN 301 199 (V1.2.1): "Digital Video Broadcasting (DVB); Interaction channel for Local Multi-point Distribution Systems (LMDS)".

[9] ETSI ETS 300 800 (1998): "Digital Video Broadcasting (DVB); Interaction channel for Cable TV distribution systems (CATV)".

[10] ETSI ETS 300 801 (1997): "Digital Video Broadcasting (DVB); Interaction channel through Public Switched Telecommunications Network (PSTN)/ Integrated Services Digital Networks (ISDN)".

[11] ETSI ETS 300 802 (1997): "Digital Video Broadcasting (DVB); Network-independent protocols for DVB interactive services".

[12] ISO 10646-1: "Information technology - Universal multiple-octet coded character set (UCS), Part 1: Architecture and Basic Multilingual Plane".

[13] ISO 639-2: "Codes for the representation of names of languages - Part 2: Alpha-3 code".

[14] ISO 8859-1: "Information technology -- 8-bit single-byte coded graphic character sets -- Part 1: Latin alphabet No. 1".

[15] ISO/IEC 13818-1: "Information technology - Generic coding of moving pictures and associated audio information: Systems".

[16] ISO/IEC 13818-6 (1998): "Information technology - Generic coding of moving pictures and associated audio information - Part 6: Extensions for DSM-CC".

[17] IETF RFC 2616 (1999): "Hypertext Transfer Protocol -- HTTP/1.1".

[18] IETF RFC 2396 (1998): "Uniform Resource Identifiers (URI): Generic Syntax".

[19] IETF RFC 1112 (1989): "Host extensions for IP multicasting".

[20] ETSI EN 301 790 (V1.2.2): "Digital Video Broadcasting (DVB); Interaction channel for satellite distribution systems".

[21] HAVi, 1.1 , This comprises the following documents:
HAVi v1.1 Chapter 8, 15-May-2001
HAVi v1.1 Java L2 APIs, 15-May-2001
HAVi v1.1 Chapter 7,15-May-2001

NOTE: Available at http://www.havi.org.

[22] Java TV JSR927: Java TV.

NOTE: Available at API http://www.jcp.org.

[23] ITU-T Recommendation X.680: "Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation".

[24] IETF RFC 2246 (1999): "The TLS Protocol, Version 1.0".

[25] IETF RFC 2045 (1996): "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies".

[26] XML 1.0, Second Edition: "Extensible Markup Language (XML) 1.0".

NOTE: Available at http://www.w3.org/TR/2000/REC-xml-20001006.

[27] IETF RFC 1035 (1987): "Domain names implementation and specification".

[28] IETF RFC 1950 (1996): "ZLIB Compressed Data Format Specification version 3.3".

[29] IETF RFC 1951 (1996): "DEFLATE Compressed Data Format Specification version 1.3".

[30] ZIP, 2003, ATSC A/100-5: "DASE-1 ZIP Archive Resource Format".

[31] IETF RFC 1945 (1996): "Hypertext Transfer Protocol -- HTTP/1.0".

[32] Modularization (2001): "Modularization of XHTML".

NOTE: Available at http://www.w3.org/TR/xhtml-modularization.

[33] ECMA-262: "ECMAScript Language Specification".

NOTE: Available at http://www.ecma-international.org/publications/standards/Ecma-262.htm.

[34] Document Object Model (DOM) Level 2 Core Specification, V1.0, W3C Recommendation 13 November, 2000.

NOTE: Available at http://www.w3.org/TR/DOM-Level-2-Core.

[35] Document Object Model (DOM) Level 1 Specification, V1.0, W3C Recommendation 1 October, 1998.

NOTE: Available at http://www.w3.org/TR/REC-DOM-Level-1.

[36] Document Object Model (DOM) Level 2 Events Specification, V1.0, W3C Recommendation 13 November, 2000.

NOTE: Available at http://www.w3.org/TR/DOM-Level-2-Events.

[37] Document Object Model (DOM) Level 2 Style Specification, V1.0, W3C Recommendation 13 November, 2000.

NOTE: Available at http://www.w3.org/TR/DOM-Level-2-Style.

[38] Document Object Model (DOM) Level 2 Views Specification, V1.0, W3C Recommendation 13 November, 2000.

NOTE: Available at http://www.w3.org/TR/DOM-Level-2-Views.

[39] CSS 2, 1998, "Cascading Style Sheets, level 2 CSS2 Specification", Including the errata published in REC-CSS2-19980512.

NOTE: Available at http://www.w3.org/TR/REC-CSS2/.

[40] Namespaces in XML, 1999, World Wide Web Consortium 14-January-1999.

NOTE: Available at http://www.w3.org/TR/REC-xml-names.

[41] HTML 4, 4.01: "HTML 4.01 Specification", W3C Recommendation 24 December 1999.

NOTE: Available at http://www.w3.org/TR/html401.

[42] SVG, 1.0: "Scalable Vector Graphics (SVG) 1.0 Specification", W3C Candidate Recommendation 2 August 2000.

NOTE: Available at http://www.w3.org/TR/2000/CR-SVG-20000802/.

[43] IETF RFC 2109 (1997): "HTTP State Management Mechanism".

[44] IETF RFC 1123 (1989): "Requirements for Internet Hosts Application and Support".

[45] IETF RFC 2818 (2000): "HTTP over TLS".

[46] Associating Style Sheets with XML documents, 1.0, W3C Recommendation 29 June 1999.

NOTE: Available at http://www.w3.org/TR/xml-stylesheet.

[47] PBP 1.1: "Personal Basis Profile 1.1 for the for the J2ME Platform (JSR 217)".

NOTE: Available at http://www.jcp.org/.

[48] ETSI TS 102 823: "DVB Specification for the carriage of synchronized auxiliary data in DVB transport streams".

[49] OCAP 1.1: "OpenCable Application Platform Specification OCAP 1.1 Profile".

[50] ETSI TS 102 034: "Digital Video Broadcasting (DVB); Transport of MPEG-2 Based DVB Services over IP Based Networks".

[51] ETSI TS 102 539: "Digital Video Broadcasting (DVB); Carriage of BCG information over IP".

[52] ETSI TS 102 816: "Digital Video Broadcasting (DVB); PVR/PDR Extension to the Multimedia Home Platform".

[53] IETF RFC 2838 (2000): "Uniform Resource Identifiers for Television Broadcasts".

NOTE: Available at http://www.ietf.org/rfc/rfc2838.txt?number=2838.

[54] POSIX, ISBN:1-55937-255-9, IEEE Standard 1003.2-1992: "Standard for Information Technology - Portable Operating System Interfaces (POSIX) - Part 2: Shell and Utilities".

[55] ETSI TS 101 699: "Digital Video Broadcasting (DVB); Extensions to the Common Interface Specification".

2.2 Informative references

The following referenced documents are not essential to the use of the present document but they assist the user with regard to a particular subject area. For non-specific references, the latest version of the referenced document (including any amendments) applies.

[i.1] ETSI TR 101 154 (V1.6.1): "Digital Video Broadcasting (DVB); Implementation guidelines for the use of MPEG-2 Systems, Video and Audio in satellite, cable and terrestrial broadcasting applications".

[i.2] ETSI TR 101 162: "Digital broadcasting systems for television, sound and data services; Allocation of Service Information (SI) codes for Digital Video Broadcasting (DVB) systems".

[i.3] ETSI ETR 211 (1997): "Digital Video Broadcasting (DVB); Guidelines on implementation and usage of Service Information (SI)".

[i.4] ETSI TR 101 200 (V1.1.1): "Digital Video Broadcasting (DVB); A guideline for the use of DVB specifications and standards".

[i.5] ETSI TR 101 201 (V1.1.1): "Digital Video Broadcasting (DVB); Interaction channel for Satellite Master Antenna TV (SMATV) distribution systems; Guidelines for versions based on satellite and coaxial sections".

[i.6] ETSI TR 101 202 (V1.1.1): "Digital Video Broadcasting (DVB); Implementation guidelines for Data Broadcasting".

2.3 Superseding references

MHP terminal specifications are allowed to supersede references used in GEM. For the features used by the MHP terminal specification, subsequent versions of those referenced specifications must provide features that are fully backwards compatible with the version used by GEM.

NOTE: It is the responsibility of the organization writing the MHP terminal specification that superseding references are compatible with GEM.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 102 034 [50] apply.

3.1.1 Definitions from GEM

GEM [1], clause 3.1 is included in the present document, with the following notes and modifications.

In the body of definitions only, the interpretations described in clause 4.2 are to be applied.

3.1.2 Definitions introduced by MHP

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

divergence: everything that violates an assertion in a specification and/or a conformance clause

NOTE: A divergence from the GEM specification is when a correctly written conformance test for a GEM specification assertion would fail.

DVB-HTML actor: 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

NOTE: The actor runs inside a user agent (native, plug-in or downloaded). The nature of the process is not defined explicitly as it depends on the nature of the user agent itself. More than one such locus of activity may be present in any given user agent.

DVB-HTML application: set of documents selected from the DVB-HTML family of elements and content formats as defined in the present document

NOTE: The extent of the set is described by the application boundary.

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

DVB-HTML document: complete unit of one the HTML family of elements or content formats defined in the present document

hidden service: DVB service that shall not be presented to the end-user by the navigator, either in a UI showing the list of available services or in response to the program up / program down keys on conventional remote controls

NOTE: These are typically be used for video streams that (from the point of view of the end-user) form part of a single service, where creating multiple video streams with a common PCR is impractical. As an additional measure to prevent presentation of individual video (and audio) streams within such a service by anything other than an appropriately authored MHP application, the `stream_type` `private_PES` may be used. The mechanism or mechanisms by which a service is identified as being hidden are outside the scope of the present document.

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

internet access application: application which is resident on an MHP terminal and used for presenting content from the internet, also referred to as "internet clients" and "internet client applications"

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

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 [18], 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 present document 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 connected resource: resource used as part of the MHP which, on its own, is not conformant to the present document but which is connected to an MHP Terminal in such a way that the whole is part of the MHP

MHP service: logical service in an MHP which can be selected through the service selection API or functional equivalents

NOTE: This includes broadcast DVB services, stored services and MHP applications executed in response to an AIT file loaded over the interaction channel.

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

MHP terminal: 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

trigger: event that may cause a change in the behaviour of a DVB-HTML application that registers interest in such events

NOTE: 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.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in GEM [1], clause 3.2 and the following apply:

API	Application Programming Interface
GEM	Globally Executable MHP
MHP	Multimedia Home Platform
UTC	Universal Time Coordinated

NOTE: Also expanded as Coordinated Universal Time.

4 General considerations and conventions

4.1 General considerations

4.1.1 Purpose

The Multimedia Home Platform (MHP) middleware standard defines a comprehensive platform that enables interactive television services to be deployed that are interoperable across any manufacturer's implementations of the standard. MHP is a comprehensive specification of a receiving device (an MHP terminal). MHP terminals receive digital video broadcasting services based on standards for various transmission media including satellite, cable, terrestrial, microwave, and TCP/IP. The transport layer may be DVB-T/T2, DVB-C/C2, DVB-S/S2, etc., or an IP transport.

One element of the MHP standard is a description of the terminal facilities that can be exploited by applications that form a part of a broadcast service. These facilities may be exposed via APIs (Application Programming Interfaces); such APIs carry semantic guarantees. Similarly, receiver functionality can be exposed with a declarative content format that contains semantic guarantees. Another element of the MHP standard is the specification of the terminal hardware and signalling infrastructure that allows it to be connected to any compatible network.

4.1.2 Format

The present document takes the form of a large number of normative references to the GEM specification.

4.1.3 Inclusion of MHP features

4.1.3.1 Subsetting prohibited

Specifications that reference the present document shall include it in its entirety. It is prohibited to base any specification on the present document if the referencing document does not require all normative requirements of the present document.

4.1.4 Application areas

GEM [1] defines three targets: Broadcast, Packaged Media and IPTV. Of these, MHP adopts the Broadcast and IPTV targets, but not the Packaged Media target.

The present document considers three application areas - 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. Interactive Broadcasting enables a range of interactive services associated with or independent from broadcast services; it 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 broadcasts services.

4.1.5 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 this release of the MHP specification, profiles are mapped to the above mentioned application areas.

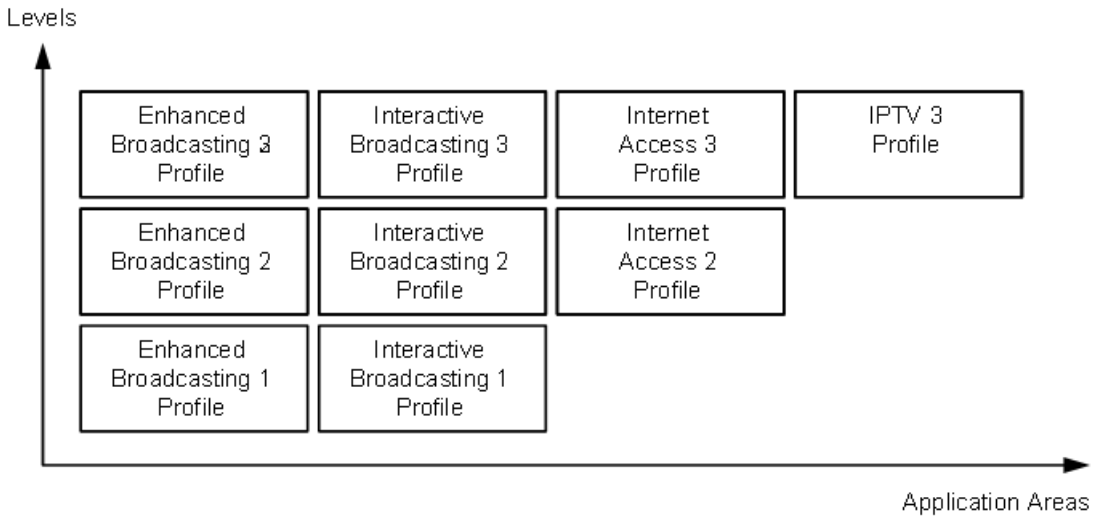

Figure 1: Application areas and levels of profiles

Figure 1 shows nine example profiles, derived from three levels for each of the four application areas. The specific definition of the profiles and the particular backward and cross compatibility between profiles is provided in clause 15, "Detailed platform profile definitions". 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.

Additionally the present document defines how the DVB Multimedia Home Platform can support delivery of DVB services over broadband IP networks. It identifies which existing MHP APIs need to interface to broadband IP networks and specifies the extended semantics of those APIs under those circumstances. Where necessary, new APIs are defined to exploit or access the features of broadband IP networks and the protocols for delivery of DVB services of those networks defined in TS 102 034 [50]. The present document also includes optional access to information carried according to TS 102 539 [51].

4.2 Conventions

GEM [1], clause 4.2 is included in the present document with the following additions and modifications.

4.2.1 References within the GEM specification

GEM [1] contains numerous internal references. In certain cases, a clause of the GEM specification that is referenced by MHP will refer to a clause of the GEM specification that is not referenced by MHP, or to a clause whose requirements are modified by MHP. In the preparation of the MHP document, every effort has been made to identify these internal references, and indicate where they do not apply or where they should be interpreted as referring to a corresponding clause of MHP.

In case of error, such internal GEM references should be interpreted as referring to the appropriate clause of MHP. That is, if MHP modifies or removes a normative requirement of GEM, for the purposes of MHP any references to that clause of the GEM specification shall be interpreted as referring to the appropriate clause of MHP, unless there is an explicit statement to the contrary in the present document.

4.2.2 Terminology in the GEM specification

4.2.2.1 GEM

The present document makes numerous references to GEM [1]. When a clause of the GEM specification is referenced from MHP, for the purposes of MHP references to GEM are to be interpreted to apply to MHP, and to MHP terminal specifications. Similarly, "GEM implementations" and "GEM terminal" are to be interpreted as "implementations of terminal specifications based on GEM," etc. "GEM application" is to be interpreted as "MHP application".

4.2.3 Inclusion of clauses of the GEM specification

Unless otherwise noted, inclusion of a clause or annex of GEM [1] implies inclusion of all clauses.

4.2.4 Conventions within the present document

Use of the term "MHP" within a normative clause of the present document shall be interpreted as referring to the present document.

4.2.5 References to OCAP

Where the words "shall be supported" or "may be supported" are used referring to a clause from OCAP [49], the following terms in the referenced clause shall be re-defined as follows:

- In the case of IPTV networks, references to the XAIT shall be interpreted as references to the signalling defined in clause 10.16.2, "XAIT".
- In the case of classical DVB networks, references to the XAIT shall be interpreted as references to the signalling defined in clause 10.17, "XAIT for classical DVB networks".

NOTE: For classical DVB networks, re-use of the XAIT descriptors from OCAP is required by clause 10.17.1, "XAIT Definition".

- References to OOB or Extended Channel MPEG section flow shall be interpreted as references to signalling defined in clause 10.17.2, "Signalling of transport via object carousel".

5 Basic Architecture

GEM [1] does not mandate a basic architecture. The informative example described in clauses 5.1 through 5.4 of GEM [1] are included as indicated in the following sections and are normative for MHP terminal specifications.

5.1 Context

GEM [1], clause 5.1 is included in the present document.

5.2 Architecture

GEM [1], clause 5.2 is included in the present document.

5.3 Interfaces Between an MHP Application and the MHP System

GEM [1], clause 5.3 is included in the present document.

5.4 Plug-ins

GEM [1], clause 5.4 is included in the present document.

5.5 IPTV in relation to previous MHP versions

Figure 2 summarizes the overall structure of the profiles defined in or derived from the present document.

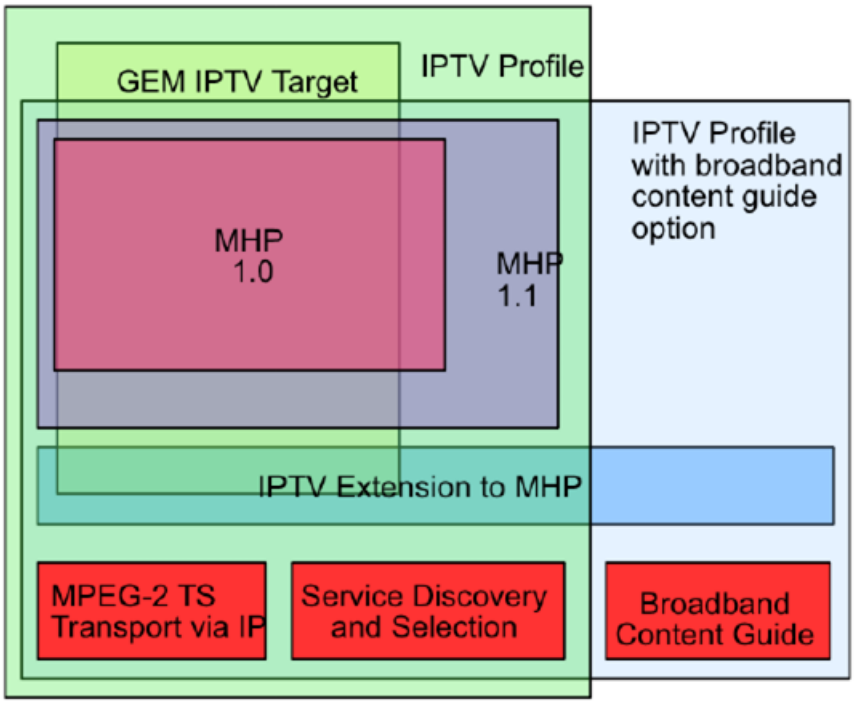

Figure 2: Structure of profiles and options

The IPTV profile is the basis of the other profiles, options and targets. Like MHP 1.0, it defines the interfaces between a MHP IPTV terminal and the network to which it is connected. The protocols needed for IPTV are fully specified based on a selection from TS 102 034 [50].

The IPTV profile can be extended with options, one of which is shown here, the broadband content guide option which integrates support for TS 102 539 [51]. Since the broadband content guide is based on TV-Anytime, the broadband content guide option includes elements of the TV-Anytime APIs defined as part of the MHP/PVR specification - TS 102 816 [52].

DVB specified protocols are not appropriate for all markets. Some markets may be using proprietary protocols and may need to have MHP/IPTV applications co-exist with these protocols. Some markets may not use the DVB-SI concepts which form the basis of the DVB IPTV protocols and never will. The GEM IPTV target defines a subset of the IPTV profile which can be used under both these circumstances. The GEM IPTV target is defined in a separate document (GEM [1]).

Not shown in the figure above is the privileged application option. This is disconnected from any specific details of IPTV. This includes features which may be appropriate in MHP terminals purchased by a subsidizing service provider. Where this option is supported, it is in turn optional for operators to exercise this control. Where one of these facilities gives an operator control over some aspect of the operation of an MHP terminal (e.g. resource management), the MHP terminal implementation shall be prepared for the subsidizing service provider not to exercise that control. Hence the MHP terminal must include its own implementation of that feature if appropriate. The support for privileged applications is closely modelled on that in OCAP [49]. An overview of the capabilities of these in OCAP can be found in clause 21.2.1 of that document. The present document requires some (but not all) of these capabilities and defines new capabilities itself.

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 [11] 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 [16] 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 [9], ETS 300 801 [10], EN 301 193 [6], EN 301 195 [7], EN 301 199 [8], TR 101 201 [i.5], EN 301 790 [20] respectively for CATV, PSTN/ISDN, DECT, GSM, LMDS, SMATV and satellite networks. The network dependent protocols work together with the Network Independent Protocols.

6.2 Broadcast Channel Protocols

This clause deals with the DVB defined or referenced broadcast channel protocols. This clause 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 annex H, "Extensions".

GEM [1], clause 6.2 is included in the present document. Clauses are included as indicated with the following notes and modifications:

Broadcast protocol support that is optional (informative) in GEM [1] is required (normative) for MHP terminals.

Figure 3 illustrates the set of DVB defined broadcast protocols that are accessible by MHP applications in some or all profiles (see clause 15, "Detailed platform profile definitions"). The full details of the APIs that provide access to these broadcast protocols are in clause 11, "DVB-J Platform".

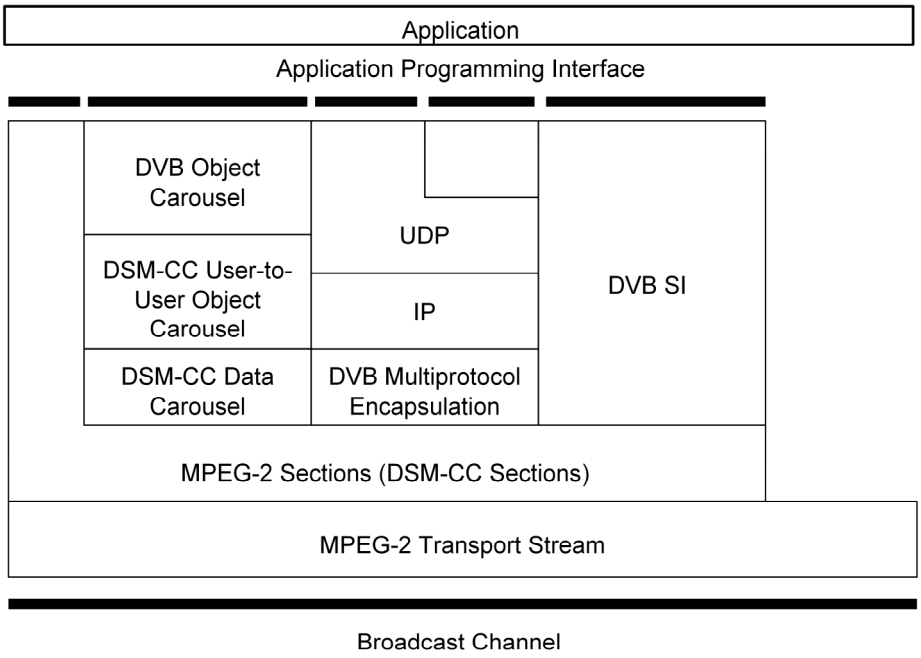

Figure 3: Broadcast Channel Protocol Stack

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

6.2.1 MPEG-2 Transport Stream

GEM [1], clause 6.2.1 is included in the present document.

6.2.2 MPEG-2 Sections

GEM [1], clause 6.2.2 is included in the present document.

6.2.3 DSM-CC Private Data

GEM [1], clause 6.2.3 is included in the present document.

6.2.4 DSM-CC Data Carousel

GEM [1], clause 6.2.4 is included in the present document.

6.2.5 Object Carousel

GEM [1], clause 6.2.5 is included in the present document with the following modifications:

MHP-compliant terminals shall implement the DSM-CC User-to-User Object Carousel protocols as defined in ISO/IEC 13818-6 [16] with the restrictions and extensions as defined in EN 301 192 [5], TR 101 202 [i.6] and annex B, "Object carousel". The present document does not permit substitution of a functional equivalent.

6.2.6 DVB Multiprotocol Encapsulation

GEM [1], clause 6.2.6 is included in the present document with the following modifications:

- This clause, which is informative in GEM [1], is normative in the present document.

6.2.7 Internet Protocol (IP)

GEM [1], clause 6.2.7 is included in the present document.

6.2.8 User Datagram Protocol (UDP)

GEM [1], clause 6.2.8 is included in the present document.

6.2.9 DVB Service Information

GEM [1], clause 6.2.9 is replaced with the following:

- DVB Service Information as defined in EN 300 468 [4] and ETR 211 [i.3].

6.2.10 IP signalling

GEM [1], clause 6.2.10 is replaced with the following:

- IP Notification Table (INT) as defined in EN 301 192 [5].

6.3 Interaction Channel Protocols

GEM [1], clause 6.3 is included in the present document, with the following notes and modifications. For this clause, only the explicitly indicated subclauses are included.

Additional protocols are required by the present document. The diagram is replaced with the following one.

Figure 4 illustrates the set of DVB defined interaction channel protocols that are accessible by MHP applications in some or all profiles (see clause 15, "Detailed platform profile definitions"). The full details of the APIs that provide access to these interaction protocols are in clause 11, "DVB-J Platform".

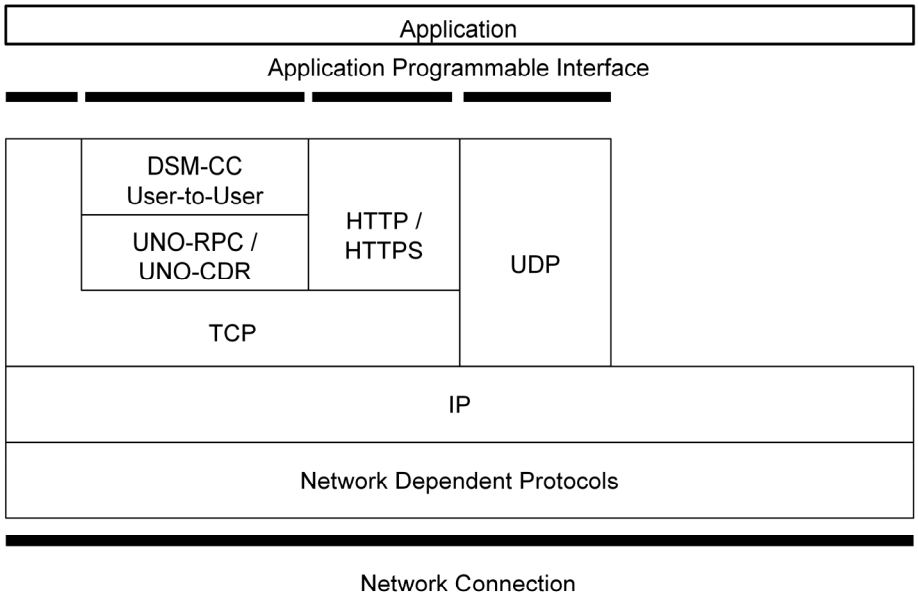

Figure 4: Interaction Channel Protocol Stack

6.3.1 Network Dependent Protocols

GEM [1], clause 6.3.1 is included in the present document.

6.3.2 Internet Protocol (IP)

GEM [1], clause 6.3.2 is included in the present document.

6.3.3 Transmission Control Protocol (TCP)

GEM [1], clause 6.3.3 is included in the present document.

6.3.4 UNO-RPC

GEM [1], clause 6.3.4 is included in the present document.

6.3.5 UNO-CDR

GEM [1], clause 6.3.5 is included in the present document.

6.3.6 DCM-CC User to User

GEM [1], clause 6.3.5 is included in the present document.

6.3.7 HyperText Transfer Protocol (HTTP)

6.3.7.1 HTTP 1.1

GEM [1], clause 6.3.7.1 is included in the present document with the following modifications:

- HTTP 1.1 support is required in all MHP profiles and targets.

6.3.7.2 MHP profile of HTTP 1.0

GEM [1], clause 6.3.7.2 is included in the present document.

6.3.7.3 HTTPS

GEM [1], clause 6.3.7.3 is included in the present document.

6.3.8 User Datagram Protocol (UDP)

GEM [1], clause 6.3.8 is included in the present document.

6.3.9 DNS

GEM [1], clause 6.3.9 is included in the present document.

6.3.10 Additional Transport Protocols

GEM [1], clause 6.3.10 is included in the present document.

6.4 Transport protocols for application loading over the interaction channel

GEM [1] clause 6.4 is included in the present document.

6.5 IPTV protocols

GEM [1], clause 6.5 is included in the present document.

7 Content formats

7.1 Static formats

7.1.1 Bitmap image formats

GEM [1], clause 7.1.1 is included in the present document.

7.1.2 MPEG-2 I-Frames

GEM [1], clause 7.1.2 is included in the present document.

7.1.3 MPEG-2 Video "drips"

GEM [1], clause 7.1.3 is included in the present document.

7.1.4 Monomedia format for audio clips

GEM [1], clause 7.1.4 is included in the present document with the following modifications:

- The last paragraph of the clause explaining the cases in which MPEG-1 is not required does not apply to the present document.

7.1.5 Monomedia format for text

GEM [1], clause 7.1.5 is included in the present document.

7.2 Broadcast streaming formats

7.2.1 Audio

MPEG Audio with the restrictions and enhancements defined in TR 101 154 [i.1].

7.2.2 Video

MHP terminals for 625-line/50 Hz markets shall support video as defined in clause 5.1 of TR 101 154 [i.1].

MHP terminals for 525-line/60 Hz markets shall support video as defined in clause 5.3 of TR 101 154 [i.1].

If high definition MPEG-2 video is supported then it shall be supported as defined in clause 5.2 or 5.4 of TR 101 154 [i.1]. If high definition MPEG-4/AVC video is supported then it shall be supported as defined in clause 5.7 of TR 101 154 [i.1].

7.2.3 Subtitles

GEM [1], clause 7.2.3 is included in the present document with the following modifications:

- Support for DVB subtitles and teletext subtitles is required in the present document.

7.3 Resident fonts

GEM [1], clause 7.3 is included in the present document with the following notes and modifications:

- Resident font support is required (normative) for MHP terminals.

7.4 Downloadable Fonts

7.4.1 PFR

GEM [1], clause 7.4.1 is included in the present document, with the following notes and modifications:

- The PFR downloadable font format as defined by GEM [1], clause 7.4.1 is required (normative) for MHP terminals.

7.4.2 OpenType

GEM [1], clause 7.4.2 is included in the present document.

7.5 Colour Representation

GEM [1], clause 7.5 is included in the present document.

7.6 MIME Types

GEM [1], clause 7.6 is included in the present document, with the following modifications:

- MHP terminal specifications may not replace the entry "application/dvb.pfr".
- Support for "image/dvb.subtitle", "text/dvb.subtitle", "text/dvb.teletext" and "multipart/dvb.service" is required by the present document.
- Additional row on table 7.

MIME type	Extension	Definition of content
"application/xml"	".xml"	As defined in clause 8 "DVB-HTML"

8 DVB-HTML

NOTE: The contents of this clause may be substantially replaced in a later release of the present document. Readers are encouraged to check for the existence of such a later release before proceeding.

8.1 Introduction

8.1.1 Application Area

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

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

- Definition of the term DVB HTML application and its lifecycle in clause 9.3.1, "The DVB-HTML Application".
- How to signal a DVB HTML application in clause 10, "Application Signalling".
- The definition of content for DVB HTML in this clause.

8.1.2 Profiles

There is only one DVB-HTML profile defined see clause 15, "Detailed platform profile definitions".

8.2 Architecture

8.2.1 Context

A DVB-HTML application in the context of MHP is a set of resources selected from the DVB-HTML family of content formats as defined in the present document. The resources of a DVB-HTML application form a directed graph, rooted by the resource referenced in the Application Entry Point (see clause 10.10.2, "DVB-HTML application location descriptor"). The allowed extent of the graph is described by the application boundary descriptor, the nodes of the graph are the physical representation of the resources, the arcs are naming references defined in the content formats. The interpretation of a DVB-HTML application is handled by a support application (commonly called a user agent in W3C terminology). For the purposes of exposition DVB defines an "actor" which is the runtime context associated with a DVB-HTML application maintained by the user agent, there is a one to one correspondence between actors and running applications. There may be a many to one relationship between actors and user agents. The lifecycle of a DVB-HTML application is described in clause 9.3, "DVB-HTML Model".

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

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

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

8.2.2 Integration Aspects

8.2.2.1 Accessing DVB-J from ECMAScript

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

8.2.2.2 Implementation of user agents via plug-ins

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

- clause 5, "Basic Architecture";
- clause 9.8, "Plug-ins";
- clause 10, "Application Signalling":
  - clause 10.13, "Plug-in signalling";
- clause 12, "Security":
  - clause 12.13, "Plug-ins".

8.3 Application Format

8.3.1 Basic Considerations

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

In particular the following technologies are used:

- XHTML (see clause 8.5).
- CSS (see clause 8.8).
- DOM (see clause 8.11).
- ECMAScript (see clause 8.10).
- XML (see clause 8.4).

8.3.2 Approach to Subsetting

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

8.4 XML

XML is a license-free, platform-independent and well-supported method for structuring data in textual form. The DVB-HTML language as defined in clause 8.5, "DVB Mark-up Language (DVB-HTML)" conforms to XML 1.0 [26]. Since its creation XML has become the foundation of new internet mark-up languages specified by the W3C.

The grammar of the DVB-HTML language is described in the DTD (Document Type Definition) in annex AA. This DTD conforms to the XHTML Host Language Document Type conformance clauses defined in the XHTML modularization specification Modularization [32].

8.5 DVB Mark-up Language (DVB-HTML)

DVB Mark-up language is an application of XML. The XML document character set for DVB-HTML shall be either the UTF-8 or UTF-16 transformation format of the Universal Multiple-Octet Coded Character Set (UCS) ISO 10646-1 [12]. The set of characters that shall be displayable is defined in annex E, "Character set".

8.5.1 Conformance considerations

8.5.1.1 Document conformance

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

- A valid document against the DVB-HTML DTD.
- An invalid document made conformant by the clauses in clause 8.5.1.1.2, "Invalid but conformant documents".

DVB HTML documents shall either not include a standalone declaration, or shall use the value "no".

8.5.1.1.1 General rules

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

A DVB HTML conformant document should include an XML declaration.

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

a) The root element shall be <html>.

b) The root element of the document shall designate the XHTML namespace using the xmlns attribute defined in Namespaces in XML [40]. The namespace designator for XHTML is "http://www.w3.org/1999/xhtml".

c) There shall be a DOCTYPE declaration in the document prior to the root element. The public identifier included in the DOCTYPE declaration shall be present and should reference the DTD found in annex AA using its Formal Public Identifier (FPI).

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

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

   Where values of x and y are found in table 1 for this version of the DVB-HTML 1.0 DTD.

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

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

Where values of x and y are found in table 1 for this version of the DVB-HTML 1.0 DTD:

Table 1: Version identifiers

	x	y
value	1	0

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

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

8.5.1.1.2 Invalid but conformant documents

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

- An invalid document with respect to the DTD found in annex AA and for which invalidity is only due to the presence of additional elements and attributes in a different namespace to the XHTML or DVB-HTML namespaces, and which does not define a new default namespace (see Namespaces in XML [40]).

- An invalid document with respect to the DTD found in annex AA with an FPI in the document type declaration identifying a strictly later version of DVB-HTML than that defined in the present document and with additional elements or attributes in the XHTML or DVB-HTML namespaces, which, when removed, results in a document that would validate against the DTD found in annex AA.

A non-conformant document is one which is not valid with respect to the DTD found in annex AA, and is not made conformant by the above clauses.

Handling of non-conformant DVB-HTML documents is specified in clause 8.5.1.2.1, "Error handling".

A document that is DVB-HTML conformant but not valid with respect to the DTD found in annex AA, is handled as detailed in clause 8.5.1.2.2, "Handling of invalid but conformant documents".

8.5.1.2 DVB-HTML user agent conformance

The DVB-HTML user agent shall be conformant with the XHTML Family User Agent conformance clauses defined in Modularization [32].

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

- Any received XML document signalled in the AIT as being part of the DVB-HTML application.

- Any XML document whose document type declaration contains one of the DVB-HTML FPIs (see clause 8.5.1.1, "Document conformance").

For the purpose of validation, it shall:

- Remove from the XML document any elements and attributes in a different namespace than the XHTML or DVB-HTML namespaces.

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

It shall subsequently validate (see clause 5.1 in XML 1.0 [26]) the resulting XML document against the DTD found in annex AA. Validation is performed by checking that the document does not violate any validity constraints as listed in table 2. Any violation report indicates the non conformance of the received XML document. Corresponding error handling is described in clause 8.5.1.2.1, "Error handling".

The way this process is performed is implementation dependent.

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

The DVB-HTML user agent is not required to be a full validating processor (according to W3C terminology in XML 1.0 [26]), in the sense that it is not required to be able to read and process any arbitrary DTDs and external parsed entities referenced in the document. It is also not required to handle an internal DTD subset attached to the document type declaration. For that reason, only the following XML validity constraints (see XML 1.0 [26]) are required to be checked.

Table 2: List of applicable validity constraints

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

8.5.1.2.1 Error handling

The user agent shall not present or allow interaction with a document which is not conformant (see clause 8.5.1.1, "Document conformance"). The DVBDOMStable event in clause 8.11.2.3.2 shall not be generated for the present document.

8.5.1.2.1.1 On receipt of a document

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

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

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

8.5.1.2.1.2 On DOM mutation

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

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

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

8.5.1.2.2 Handling of invalid but conformant documents

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

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

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

8.5.2 Set of modules required by the present document

The DVB-HTML document type definitions are made up of the following abstract modules indicated in table 3. The modules are defined in page 15 of Modularization [32]. An implementation of the DTD is defined in annex AA.

Table 3: Required modules for DVB-HTML

Modularization	Required
Structure	Yes
Text	Yes
Hypertext	Yes
List	Yes
Applet	
Presentation	Yes
Edit	
Bi-directional Text	Yes
Basic Forms	
Forms	Yes
Basic Tables	Yes
Tables	
Image	Yes
Client side Image map	Yes
Server side image map	
Object	Yes
Frames	Yes
Target	Yes
Iframe	Yes
Intrinsic events	
DVB Intrinsic events	Yes
Metainformation Module	Yes
Scripting	Yes
Style sheet	Yes
Style Attribute	Yes
Link	Yes
Base	Yes
Name Identification	
Legacy	

8.5.3 Semantics for modules

8.5.3.1 XHTML modules

For the modules defined by Modularization [32] the semantics are as defined there except as elaborated below and in clause 8.6, "Media Types".

8.5.3.1.1 Structure

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

8.5.3.1.2 Text

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

8.5.3.1.3 Hypertext

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

8.5.3.1.4 Presentation

NOTE: The HTML conformance clause (see HTML 4 [41]) for this module does not require text to be rendered with a different appearance when these elements are used in the absence of other styling information.

8.5.3.1.5 Forms

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

NOTE: An implementation is not required to support uploading of local files, therefore where input type="file" appears in a DVB-HTML document, the implementation may substitute input type="hidden".

8.5.3.1.6 Client-side Image Map

Semantics on the way to associate an image map with an element are defined in section 13.6.1 in HTML 4 [41] with the following exception:

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

If no `<map>` element is identified through the `usemap` attribute, then no association shall be assumed. This is in line with the module DTD implementation as defined in Modularization [32]. In HTML 4 [41] this association was performed using the `name` attribute of the `<map>` element.

8.5.3.1.7 Image

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

8.5.3.1.8 Object

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

When the referenced object is an MPEG stream, see clause 8.6.7, "MPEG Audio" and clause 8.6.8, "MPEG Video".

When the referenced object is a DVB service, see clause 8.6.9, "DVB Services".

When the referenced object is an Xlet see clause 8.9, "Xlet integration".

For other media types, see clause 8.6.10, "Graphics content".

8.5.3.1.9 Frames

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

8.5.3.1.10 Target

The semantics of specifying target frame information is described in section 16.3 in HTML 4 [41] with the following exception:

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

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

NOTE: In HTML 4 [41], this was performed through the `name` attribute. Appendix B.8 in HTML 4 [41] is not relevant any more.

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

8.5.3.1.11 Iframes

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

8.5.3.1.12 Metainformation

See clause 8.10.2.1, "ECMAScript APIs for accessing DVB-J".

8.5.3.1.13 Scripting

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

8.5.3.1.14 Link

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

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

Stylesheet: Refers to an external style sheet.

Style Sheets can also be associated with a DVB-HTML document by using a processing instruction whose target is `xml-stylesheet` as defined in Associating Style Sheets with XML documents [46]. This processing instruction follows the behaviour of the `<link rel="stylesheet">` semantics.

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

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

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

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

8.5.3.1.15 Base

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

8.5.3.2 XHTML attributes

8.5.3.2.1 Longdesc, alt and cite attributes

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

8.5.3.2.2 Accesskey attribute

The `accesskey` attribute of HTML 4 [41] assigns an access key to an element. An access key is a single character from the document character set, which if pressed gives focus to the element, the semantics of giving focus to an element are element specific (see HTML 4 [41] section 17.11.2).

In order to have some measure of compatibility with the JAVA AWT PBP 1.1 [47] and HAVi HAVi [21] event sets, the DTD used by the user agent points to an implementation specific DVB character entities module which shall define a set of XML entities which map all of the JAVA AWT PBP 1.1 [47] and HAVi HAVi [21] "`VK_`" events onto implementation specific characters.

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

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

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

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

It is not required for implementations to generate "`VK_`" events other than the minimum set specified in GEM [1] table 86, "Minimum set of input events". But the user agent shall allow documents to refer to any of them. Generation of the "`VK_`" events is allowed to be contextual, for example, when a text area element has the focus the "`VK_0`" event may not be generatable by the user, since the device specific operation may generate a "0" character instead.

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

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

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

8.5.3.3 DVB-HTML modules

8.5.3.3.1 DVB Intrinsic events

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

Table 4: Elements of DVB Intrinsic events

Elements	Attributes
body&	onload (Script), onunload (Script) (see clause 8.10 "Scripting"). ondvbdomstable (Script) (see clause 8.11.2.3.2, "DVBDOMStable event").
frame&	onload (Script), onunload (Script) (see clause 8.10 "Scripting"). ondvbdomstable (Script) (see clause 8.11.2.3.2, "DVBDOMStable event").

If these attributes are used then they must be within the scope of a namespace definition, namely "`http://www.dvb.org/mhp`". The namespace prefix shall be declared on the element of the document that uses them. And the namespace shall be "dvbhtml" (see Namespaces in XML [40]).

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

```
<?xml version="1.0"?>
<!DOCTYPE html
     PUBLIC "-// DVB//DTD XHTML DVB-HTML 1.0//EN"
     "http://www.dvb.org/mhp/dtd/dvbhtml-1-0.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >

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

8.6 Media Types

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

Table 5: Content types accessible in DVB-HTML

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

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

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

8.6.1 Uses of MIME media types

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

Table 6: Use of ContentType attribute on elements

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

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

Table 7: Use of URI attribute on elements

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

8.6.2 MIME media type use restrictions

The behaviour of the MHP terminal is only specified where there is an "X" at the intersection of element.attribute and MIME media type in table 8.

Table 8: Use of MIME media type by element

Element. attribute	MIME media type														
	text/xml	application/ xml	text/css	text/plain	text/dvb.utf8	audio/mpeg	image/jpeg	image/png	image/gif	image/mpeg	video/mpeg	multipart /dvb.service	application/ dvbj	video/dvb.mp eg.drip	text/ ecmascript
As root element	X														
a.type	X					X						X	X		
link.type (see note)	X		X												
object.type	X					X	X	X	X	X	X	X		X	
object.codetype													X		
param.type	(see note)														
style.type			X												
script.type															X
NOTE: Informative, authoring guideline: MIME media type constraints only apply to this attribute when the param.valuetype attribute has value "ref". No allowed object types required to be supported by the present document make use of this "type" attribute.															

Table 9 shows whether the resource identified by the URI value of each element.attribute pair (vertical axis) has DVB defined semantics for each MIME media type (horizontal axis) in DVB-HTML; "X" = Yes, blank = No.

Table 9: MIME media type usage

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

NOTE 1: MIME type is irrelevant here as it's just the URI that matters, not the thing it identifies.
NOTE 2: Specific to the object.

8.6.3 Semantics of media type

a) See clause 8.16.5.2, "Exit locator".

b) See clause 8.6.7, "MPEG Audio".

c) See clause 8.6.8, "MPEG Video".

d) See clause 8.6.5, "Application content".

e) See clause 8.6.9, "DVB Services".

f) See clause 8.5.3.2.1, "Longdesc, alt and cite attributes".

g) See clause 8.6.4, "Frame content".

h) See clause 8.6.10, "Graphics content".

i) See clause 8.6.11, "Script content".

j) See clause 8.6.12, "Style sheet content".

8.6.4 Frame content

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

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

8.6.5 Application content

8.6.5.1 When referenced via an AIT locator

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

EXAMPLE:

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

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

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

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

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

8.6.5.2 When not referenced via an AIT locator

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

href: The semantics are as defined in HTML 4 [41].

usemap: The referenced element defines a map to overlay on the image. The semantics are as defined in HTML 4 [41].

object.data: The referenced document is rendered within the object as if it were a frame. The semantics are as defined in HTML 4 [41].

src: The referenced document is rendered within a frame or iframe. The semantics are as defined in HTML 4 [41].

8.6.6 Relative linking

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

For `link.rel="stylesheet"`, only the type `"text/css"` is defined and the user agent shall interpret this as specified in clause 14.3.2, "Specifying external style sheets" of HTML 4 [41].

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

8.6.7 MPEG Audio

DVB HTML shall be able to reference MPEG audio as indicated in table 9. The referenced audio may be of either indefinite (see clause 7.2.1, "Audio") or definite duration (see clause 7.1.4, "Monomedia format for audio clips").

8.6.7.1 Resources of indefinite duration

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

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

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

8.6.7.1.1 Relation to document events

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

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

8.6.7.2 Resources of definite duration

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

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

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

Table 9a

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

8.6.7.2.1 Relation to document events

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

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

8.6.8 MPEG Video

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

8.6.8.1 Video Resources of indefinite duration

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

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

Video referenced by an `<object>`, `<img>` or `<input>` is considered to be referenced later than that in external style sheets, and thus takes precedence over the selected background video in the case that there are insufficient resources to display both the referenced video and the background video simultaneously (in accordance with GEM [1] clause 11.2.9 "Intra application media resource management"). In the case of multiple references within a document, it is implementation specific which is considered referenced later.

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

8.6.8.1.1 Relation to document events

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

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

8.6.8.1.1.1 Modification of the referencing attribute using the DOM

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

8.6.8.2 Resources of definite duration

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

8.6.9 DVB Services

Use of URLs with that reference a service in contexts that do not require user action (i.e `object.data`, `object.archive`) on objects of type "`multipart/dvb.service`" are defined only for the media aspects of a service (e.g. video, audio, subtitles) and shall not cause service selection, but the media components of that service shall be displayed according to (clause 8.6.7, "MPEG Audio" and clause 8.6.8, "MPEG Video").

Links to DVB services in contexts that require user action (i.e. `a.href, area.href`) shall cause service selection to the referenced service if the application is authorized (see clause 8.15.1.9, "javax.tv.service.selection.ServiceContextPermission"). If authorization fails, then the link shall be treated as if referencing an unavailable resource.

Links to DVB services of the form `"dvb:"` referencing broadcast services, and other locators referencing stored services (see GEM [1], clause 9.6.2 "Stored services") or services over the interaction channel shall be allowed (see clause GEM [1], clause 9.6.1 "Applications loaded from the interaction channel").

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

8.6.10 Graphics content

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

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

8.6.11 Script content

The referenced resource shall be in a plain text format admitted by the present document clause 7.1.5, "Monomedia format for text". The content shall be ECMAScript source code as defined in ECMA-262 [33], and shall be handled as if it were inline.

8.6.12 Style sheet content

The referenced resource shall be in a plain text format admitted by the present document clause 7.1.5, "Monomedia format for text". The content shall be CSS source code as defined in CSS 2 [39], and shall be handled as defined by the cascading rules therein.

8.6.13 HTTP(S) URLs

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

8.6.14 CSS Properties

8.6.14.1 Sources of MIME media type use points

<uri>: The CSS 2 specification defines the value type `<uri>`. For the places where `<uri>` is a permitted value in CSS see CSS 2 [39].

8.6.14.2 MIME media type use restrictions

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

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

Table 10 shows whether the resource identified by the URI value of each CSS attribute (vertical axis) has DVB defined semantics for each MIME media type (horizontal axis) in DVB-HTML; "X" = Yes, blank = No.

Table 10

CSS Attribute		MIME media Type									
		text/plain	text/dvb.utf8	image/jpeg	image/png	image/gif	image/mpeg	video/mpeg	video/dvb.mpeg.drip	multipart/dvb.service	application/dvb.pfr
background-image				x[b]	x[b]	x[b]	x[b]	x[c]	x[c]		
content		x[a]									
list-style-image				x[b]	x[b]	x[b]	x[b]				
@font-face	src										x
@viewport	"background:"			x[b]	x[b]	x[b]	x[b]		x[c]		
	"background-video-n"							x[c]		x[d]	

a) See clause 8.6.15, "Generated Content".

b) See clause 8.6.16, "Graphics styling".

c) See clause 8.6.17, "Video Styling".

d) See clause 8.6.18, "DVB Service styling".

8.6.15 Generated Content

The referenced resource shall be in a plain text format admitted by the present document clause 7.1.5, "Monomedia format for text". The content shall be handled as if it were inline as defined in CSS 2 [39].

NOTE: Informative, authoring guideline, in CSS2 you cannot:

- Generate markup (only XML CDATA).
- Apply further style to generated content.
- Access generated content via DOM.

8.6.16 Graphics styling

Graphics formats shall be supported as specified in clause 7.1.1, "Bitmap image formats". Such graphics shall be handled as defined in CSS 2 [39].

8.6.17 Video Styling

Video and video drips may be used as the `background-image style` property. The resource shall be handled as an image type as defined in CSS 2 [39]. Video is displayed as defined in clause 8.6.8.1, "Video Resources of indefinite duration" as a reference that does not require user interaction.

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

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

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

8.6.18 DVB Service styling

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

8.7 Synchronization

8.7.1 Triggers Overview

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

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

Application authors can think in terms of traditional media time bases (e.g. SMPTE time codes) that are a simple frames/seconds offset from the beginning of the media. Triggers can signal that a point on the time base has been reached. The author has to "name" these events so that applications can subscribe to them. This name may be meaningful to the author as is illustrated in table 11.

Table 11: Example trigger names

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

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

The following issues are described in more detail below:

- How triggers are transported (see clause 8.7.1.1).
- How an application registers for and receives triggers (see clause 8.7.1.2).

8.7.1.1 Transport of triggers

We consider here transport of triggers in:

- DSM-CC stream events.

8.7.1.2 Application registration and reception

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

- Explicit registration using the DOM API

Delivery is by DOM event. Trigger DOM event interfaces inherit from the interface `Event` in DOM (see Document Object Model (DOM) Level 2 Events Specification [36]).

8.7.1.3 Binding to DSM-CC Stream events

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

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

8.7.2 Trigger Events

8.7.2.1 Converting stream events into DOM events

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

a) A mechanism to override the default association between application and stream event message which provides a degree of abstraction between the transport mechanism of the stimulus aspect of the event and the author's behavioural view.

b) A mechanism which expands the media event into one or more DOM events, this includes mapping the media event name into a DOM type, and extracting any or all of the media event payload and converting it into properties of the DOM event.

The application document receives the DOM events and has its behaviour modified by them using the standard mechanism of DOM event delivery (see clause 98).

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

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

The entire process is summarized by the following data flow diagram.

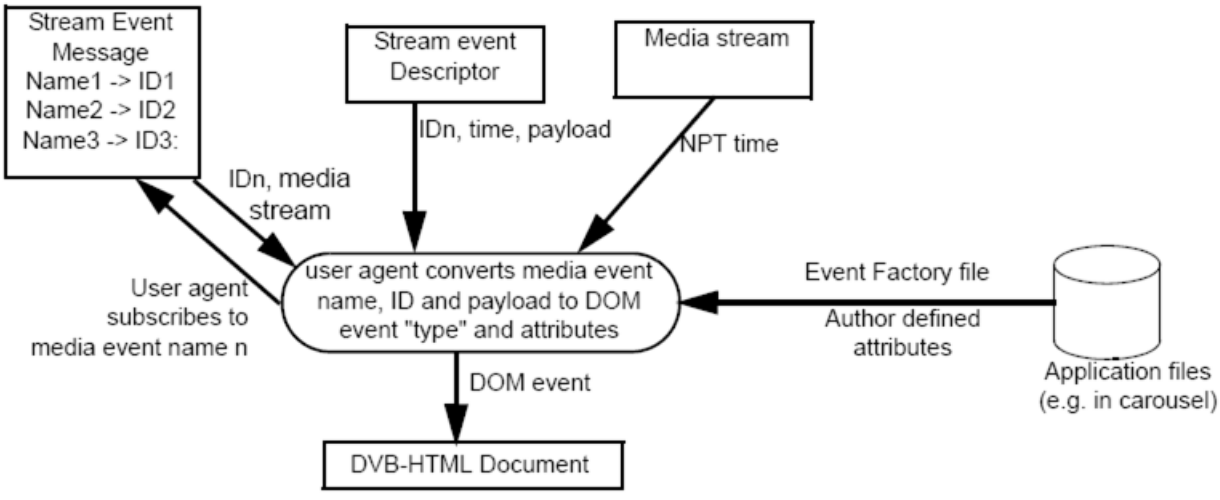

Figure 5: Overview of the authored event mechanism

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

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

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

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

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

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

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

8.7.2.2 Event Factory File definition

8.7.2.2.1 Syntax

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

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

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

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

The following URL may be used to reference this file:

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

-->

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

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

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

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

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

8.7.2.2.2 Element semantics

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

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

trigger-event: This element specifies a relationship between:

- A media stream event.
- The DOM event interface to which it maps.
- The DOM event destination.

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

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

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

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

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

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

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

8.7.2.2.3 Attributes semantics

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

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

event: The name of the media event in the stream event message which provides the stimulus for this event.

NOTE: Multiple broadcast-event records may reference the same media event, possibly with time offsets. This provides the names that the user agent subscribes to.

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

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

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

EXAMPLE 1:

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

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

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

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

EXAMPLE 2:

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

The MHP shall support offset referencing of at least ±12 hours.

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

bubbles: This attribute sets the DOM event attribute "bubbles" as defined in clause Document Object Model (DOM) Level 2 Events Specification [36].

cancelable: This attribute sets the DOM event attribute "cancelable" as defined in Document Object Model (DOM) Level 2 Events Specification [36].

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

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

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

where `pattern` follows the grammar in clause 15.10.1 of ECMA-262 [33], and flags are at most one each of ("g", "i" or "m"). The replacement text is arbitrary text, but may contain '$' values which are substituted by elements collected by the pattern following the rules in clause 15.5.4.10 of ECMA-262 [33].

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

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

would deliver the data:

```
"1234567"
```

8.7.2.3 Default Event Factory Element

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

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

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

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

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

8.7.2.4 Default Event Factory File

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

8.7.2.5 Worked example

An applications event factory file contains the following text:

```
<trigger-event
    stream="dvb://233a..1234/goals"
    event="goal"
    type="score"
    cancelable="false">
    <event-attribute
        attribute-name="score"
        attribute-data="/\[s(core)?:(\d+)-(\d+)/$1-$2" />
</trigger-event>
```

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

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

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

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

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

Then the resultant DOM event would be constructed using the following initialization:

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

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

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

8.7.2.6 System events

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

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

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

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

8.7.2.6.1 dvb.start event

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

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

8.7.2.6.2 dvb.page event

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

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

The following values for attribute-name are defined.

Table 12

Attribute-name	Number of occurrences
actuate	zero or one
href	one
title	zero or one

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

- "onLoad".
- "onRequest".

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

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

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

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

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

8.7.3 Binding the event factory file to the application

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

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

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

8.7.3.1 Syntax of event linkage file

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

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

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

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

The following URL may be used to reference this file:

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

-->

<!ELEMENT htmleventlinkagefile (linkage)*>

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

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

8.7.3.2 Semantics of event linkage file

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

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

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

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

URI: The URI.

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

The encoding of this attribute shall be ISO 639-2 [13].

8.7.3.3 Example

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

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

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

8.7.3.4 Name and location of linkage file

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

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

```
foo.html
```

The linkage file would be named:

```
foo.lnk
```

8.7.4 Default Trigger Mechanism

The default trigger mechanism is assumed when there is no linkage file associated with a DVB-HTML application. This default mode provides a simple event mechanism in the case where a DVB-HTML application is associated with system events or with trigger events with simple data payload. An overview is provided in figure 6, "Overview of the default trigger mechanism".

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

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

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

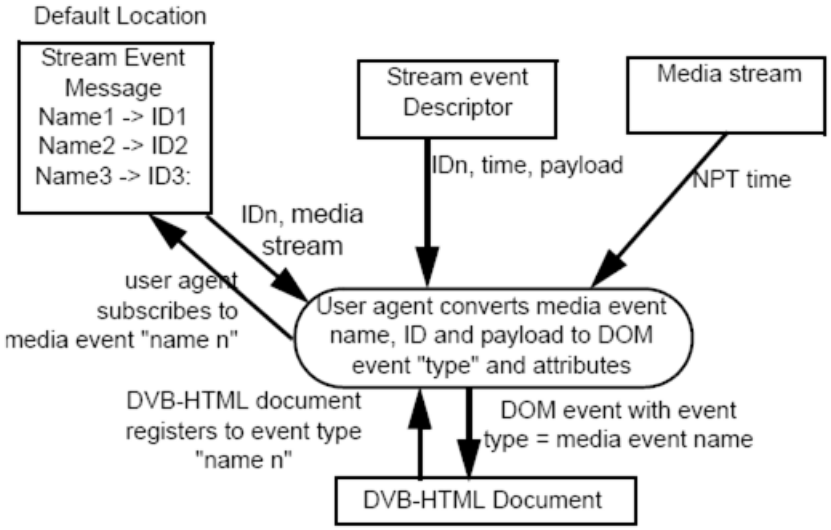

Figure 6: Overview of the default trigger mechanism

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

a) The DVB-HTML document, through an embedded Xlet or any attached script, implements an EventListener interface and subscribes to the trigger event through a call to the DOM `addEventListener()` method on the root element node of the DOM tree with an appropriate event type.

The user agent locates the DSMCC Stream Event message in the default location that contains an event name equal to the registered event type.

The DSMCC Stream Event message provides the mapping between the event name and the event id and references the source of the stream event descriptors that the user agent needs to monitor.

b) On receipt of the stream event descriptor that provides the mapping between the event id and both the eventNPT and the associated payload, the user agent begins the synthesis of the DOM event, as corresponding to the default event factory element. When the NPT of the referenced stream is equal to the eventNPT of the stream event descriptor or as soon as the stream event descriptor is received in case of 'do-it-now' events, the DOM event is injected in the DOM implementation.

c) Payload of the stream event can be fully extracted from the synthesised DOM event through the default "payload" attribute.

8.8 CSS

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

8.8.1 Summary of CSS profiling for MHP

DVB-HTML shall support CSS2 as defined in CSS 2 [39] but amended by the profile statements in the following clauses:

- Clause 8.8.2, "MHP profile of CSS data types".
- Clause 8.8.3, "MHP profile of CSS @ rules".
- Clause 8.8.4, "MHP profile of CSS media types".
- Clause 8.8.5, "Graphics and video integration".
- Clause 8.8.6, "Font selection".
- Clause 8.8.7, "Font specification".

An MHP terminal must therefore observe the conformance clauses specified in CSS 2 [39].

8.8.2 MHP profile of CSS data types

The implementations shall support all of the data types defined in CSS 2 [39] except as amended by table 13.

Table 13: MHP profile of CSS data types

Data type	Profile
`<alphavalue>`	See clause 8.8.4.2.1.1, "opacity". The results of opacity blending may be implementation specific, see clause 13.6.1, "Approximations in composition".
`<angle>`	Not required (support for aural style sheets is not required).
`<color>`	Application authors should not make any assumption on the expected results when using system colours defined in CSS 2 [39]. In particular there is no guarantee that any two system colour names will have distinct appearances.
`<frequency>`	Not required (support for aural style sheets is not required).
`<length>`	See clause 8.8.5.2, "Coordinate spaces".
`<shape>`	In the present document the only valid `<shape>` value is: `rect (<top> <right> <bottom> <left>)` The parameters <`<top>`, `<bottom>`, `<right>`, and `<left>` specify offsets from the top and left of the box.
`<time>`	Not required (support for aural style sheets is not required).

8.8.3 MHP profile of CSS @ rules

The implementations shall support all of the @ rules defined in CSS 2 [39] except as amended by table 14.

Table 14: MHP profile of CSS @ rules

Rule	Profile
page	Not required.
viewport	Required, See clause 8.8.5.3.2, "The @dvb-viewport rule".

8.8.4 MHP profile of CSS media types

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

8.8.4.1 "screen" media type

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

8.8.4.2 "dvb-tv" media type

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

The `dvb-tv` media type is a new media type introduced to enable content authors to differentiate an MHP terminal with a predominantly TV like environment (low resolution, possible interlace, specific gamut color, limited-scrollability, small screens, sound available) from an MHP terminal with a more computer desktop like environment. The `dvb-tv` media type is a superset of properties as the `screen` media type as defined in CSS 2 [39]. `dvb-tv` is a member of the following media groups.

Table 15: Relationship between media groups and media types

MediaTypes	Media Groups			
	Continuous/paged	Visual/aural/tactile	Grid/bitmap	Interactive/static
dvb-tv	continuous	visual	bitmap	both

8.8.4.2.1 Additional Properties of "dvb-tv" media type

8.8.4.2.1.1 opacity

opacity: Objects may be rendered in a semi-transparent fashion in DVB-HTML using the `opacity` property defined in SVG [42]. Opacity can be thought of as a postprocessing operation. Conceptually, after an object or group of objects is rendered into an RGBA off-screen image, the opacity setting specifies how to blend the off-screen image into the graphics device.

Value: <alphavalue> | inherit
Initial: 1.0
Applies to: all elements
Inherited: no
Percentages: N/A
Media: visual

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

See alpha compositing in clause 13.3.4 "Composition".

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

8.8.4.2.1.2 dvb-compose-rule

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

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

Value: clear | dst-in | dst-out | dst-over | src | src-in | src-out | src-over
Initial: src-over
Applies to: all elements
Inherited: no
Percentages: N/A
Media: visual

8.8.4.2.1.3 dvb-clip-video

See clause 8.8.5.8.1.1, "dvb-clip-video".

8.8.4.2.1.4 focus traversal

See clause 8.8.5.10, "Focus traversal and short-cuts".

8.8.4.2.1.5 viewports

See clause 8.8.5.3.2, "The @dvb-viewport rule".

8.8.4.2.2 Policy rules

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

8.8.4.3 Clarifications on support of paged properties

The CSS2 specification CSS 2 [39], appendix F, lists properties associated with the paged media group as being also associated with the visual media group, which creates a possible ambiguity with the definition of the `screen` and `dvb-tv` media types which do not support the `paged` media group. For clarification, DVB-MHP does not require the support of the properties where the "media group" column contains "`visual, paged`".

8.8.5 Graphics and video integration

8.8.5.1 General recap of the MHP graphics

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

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

8.8.5.1.1 Input video space

This considers post upsampling MPEG pixels.

8.8.5.1.2 Device space

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

8.8.5.1.3 Normalized space

Normalized coordinates relative to the screen.

8.8.5.1.4 Colour

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

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

8.8.5.2 Coordinate spaces

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

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

The definition of `em` and `ex` shall be computed using the rules in GEM [1], clause D.3.4.2, "Horizontal resolution", not based on the subtended angle at the eye as in CSS 2.

8.8.5.2.1 Screen coordinates

The screen space coordinates are normalized, and can therefore be expressed in CSS % units where it is defined for MHP that this means percent of screen real estate, i.e. full screen is top:0 % left:0 % width:100 % height:100 %.

8.8.5.2.2 Pixel coordinates

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

8.8.5.2.3 Video coordinates

To address positions based on the video coordinate space the *pel* numeric type is defined, e.g. the style:

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

establishes a fixed box in the viewport, the rectangle of which is based on (43, 23, 100, 60) in the source video (see clause 8.8.5.6.2, "Definition of pel areas in the video" for how this is computed into an actual box).

8.8.5.3 How to define the initial containing block

8.8.5.3.1 Problem

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

Figure 7: CSS Boxes

8.8.5.3.2 The @dvb-viewport rule

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

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

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

Specifies that:

- the viewports scene is 1/4 screen and centred;
- that the desired coordinate resolution of the viewport is 288 × 360;
- and the initial containing block is a 100 × 100 square approximately centred within the viewport.

8.8.5.3.3 Establishing a viewport

8.8.5.3.3.1 Viewport properties

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

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

Value: <shape>
Initial: rect(0%, 100%, 100%, 0%)
Applies to: Viewport context
Inherited: no
Percentages: area
Media: visual

<shape>: See table 13, "MHP profile of CSS data types". In this context `<top>, <bottom> <right>`, and `<left>` specify <percentage>.

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

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

Value: <length>
Initial: 720px
Applies to: Viewport context
Inherited: no
Percentages: N/A
Media: visual

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

Value: <length>
Initial: 576px
Applies to: Viewport context
Inherited: no
Percentages: N/A
Media: visual

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

Value: <shape>
Initial: rect(0%, 100%, 100%, 0%)
Applies to: Viewport context
Inherited: no
Percentages: Scene
Media: visual

<shape>: See table 13, "MHP profile of CSS data types". In this context `<top>, <bottom> <right>`, and `<left>` specify <length> or <percentage>.

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

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

8.8.5.3.3.2 Other viewport properties

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

These are used after the viewport is selected (see clause 8.8.5.3.4, "Pseudo classes"), and allow the application to override the presentation of the background properties.

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

Value: video | graphics | aligned | none
Initial: none
Applies to: Viewport context
Inherited: no
Percentages: N/A
Media: visual

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

Value: <shape>
Initial: rect(0%, 100%, 100%, 0%)
Applies to: Viewport context
Inherited: no
Percentages: screen
Media: visual

<shape>: See table 13, "MHP profile of CSS data types". In this context `<top>, <bottom> <right>`, and `<left>` specify <percentage>.

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

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

Value: <uri> [, <colour>] | <colour>
Initial: black
Applies to: Viewport context
Inherited: no
Percentages: N/A
Media: visual

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

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

Value: <shape>
Initial: rect(0%, 100%, 100%, 0%)
Applies to: Viewport context
Inherited: no
Percentages: screen
Media: visual

<shape>: See table 13, "MHP profile of CSS data types". In this context `<top>, <bottom> <right>`, and `<left>` specify <percentage>.

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

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

Value: <shape>
Initial: rect(0%, 100%, 100%, 0%)
Applies to: Viewport context
Inherited: no
Percentages: input video
Media: visual

<shape>: See table 13, "MHP profile of CSS data types". In this context <top>, <bottom> <right>, and <left> specify <pels> or <percentage>.

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

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

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

Value: true | false
Initial: true
Applies to: Viewport context
Inherited: no
Percentages: N/A
Media: visual

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

Value: <uri> (, <uri>)* | <color>
Initial: dvb.current
Applies to: Viewport context
Inherited: no
Percentages: N/A
Media: visual

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

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

Value: <process>
Initial: DFC_PROCESSING_DEFAULT
Applies to: Viewport context
Inherited: no
Percentages: N/A
Media: visual

<process>: One of the following constant values:

DFC_PROCESSING_CCO: A central rectangle is cut out of the video input frame and transferred into the output video rectangle defined by the corresponding background-video-rect.

DFC_PROCESSING_FULL: The full video input frame is transferred, part of the output may be black if the video input aspect is not the same as the output frame aspect.

DFC_PROCESSING_LB_14_9: The video input frame is transferred into a 14:9 letterbox in the output frame defined by the corresponding background-video-rect.

DFC_PROCESSING_LB_16_9: The video input frame is transferred into a 16:9 letterbox in the output frame defined by the corresponding background-video-rect.

DFC_PROCESSING_LB_2_21_1: The video input frame is transferred into a 2,21:1 letterbox in the output frame defined by the corresponding background-video-rect.

DFC_PROCESSING_DEFAULT: The default decoder format conversion is active.

DFC_PROCESSING_PAN_SCAN: An mxn part out of the video input frame is transferred into the output frame defined by the corresponding background-video-rect. The position of this part is determined by pan and scan vectors from the MPEG video stream.

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

8.8.5.3.3.3 n planes

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

8.8.5.3.4 Pseudo classes

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

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

The pseudo-class syntax is as follows:

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

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

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

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

resolution: specifies the physical display pixel resolution.

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

This allows most of the common formats to be defined.

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

EXAMPLE 1: 16x9R720x576P
Matches a progressive scan 16 × 9 display with 720 × 576 pixels.

EXAMPLE 2: 4x3R1024x768
Matches a 1 024 × 768 display with 4 × 3 pixel aspect ratio, either interlaced or progressive scan.

If the pseudo-class for more than one `@dvb-viewport` rule matches a display, one rule is chosen by applying the CSS 2 cascade algorithm (CSS 2 [39], section 6.4, "The cascade"). In addition to the description of specificity there, the relative specificities of several display pseudo-classes is determined by summing the number of components which are specified; for example, "P" has a specificity of 1, "16x9R720x576P" has a specificity of 3.

8.8.5.4 Cascading

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

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

8.8.5.5 How to discover where the video is

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

8.8.5.5.1 The area property

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

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

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

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

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

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

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

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

EXAMPLE 1: `area: totalVideoAreaOnScreen`
is the default, and would limit the viewport to exactly where the video is being placed (including letterbox bars).

Combinations must add up to a single rectangle.

EXAMPLE 2: `area: activeVideoAreaOnScreen & BHBar`
would be legal and would allow the use of the area in the lower letterbox bar but not the top bar.

EXAMPLE 3: **`area: LVBar & THBar`**
would be illegal.

EXAMPLE 4: `area: THBar & BHBar`
would be illegal.

EXAMPLE 5: `area: THBar & BHBar & activeVideoAreaOnScreen`
would be legal and would allow the use of the area in the upper and lower letterbox bar and the active video area.

The order of property evaluation is as follows:

- First the background video rectangle is established (see "background-video-rectangle-n", the default is full screen). The viewport is established with respect to the furthest video plane from the viewer.

- Then the `area` is determined, (the default is the total active video area). and finally the viewport and initial box are determined. These would all have to be recalculated if the background video rectangle were to change (either by the terminal or by DOM control).

8.8.5.6 Placing content in relation to video

The terminal defined starting cascade style sheet implicitly defines (see "Definition of boxes") the following regions when the viewport contains them.

```
.active-video-area-on-screen
.total-video-area-on-screen
.LV-bar
.RV-bar
.TH-bar
.BH-bar
```

so that for example the document

```
<!DOCTYPE
    html PUBLIC "-//DVB//DTD XHTML DVB-HTML 1.0//EN"
    "http://www.dvb.org/mhp/dtd/dvbhtml-1-0.dtd"
>
<html>
<head>
    <title>css demo</title>
    <style type="text/css">
    <!--
    @dvb-viewport 16x9I{
        area: active-video-area-on-screen & BH-bar
    }
    -->
</style>
</head>
    </body>
        <div class="active-video-area-on-screen"> </div>
        <div class="BH-bar"> </div>
    </body>
</html>
```

Would allow the placement of material in the lower black bar and over the video area.

8.8.5.6.1 Definition of boxes

The named regions are defined as the intersection of the viewport (shown in the illustration as the yellow rectangle) with the relevant area, therefore the regions could potentially be empty, in this case no content is displayed for that region. For example the `#TH-bar` shown in blue in the illustration below does not cover the whole of the letterbox bar, but only that which intersects the viewport. Similarly the `active-video-area-on-screen` region shown by the purple rectangle does not cover all the live video, but only that which intersects the viewport.

Figure 8: Example of named screen regions

8.8.5.6.2 Definition of pel areas in the video

Positions defined by using the pel type are defined to be the pixel location in the viewport where the corresponding video pel is translated to. In the case that the transformed pel covers more than one pixel, the top most left most pixel is used. This allows the placement of content in relation to elements within the video.

NOTE: The pixel may not be within the viewport, and so may get clipped. it could also change with pan and scan.

The video plane used to define the pel coordinate system shall be the backmost active video plane. If no video plane is active then the pel type is defined as equal to the px type.

8.8.5.7 Placing video within the presentation

8.8.5.8 Box Layout

8.8.5.8.1 Video Boxes

Video content may be placed in an `<object> or <img>` element. Using style rules the box for this can be positioned following the normal CSS rules anywhere within the viewport. This acts like a video component in DVB-J, and is entirely separate from video in the background plane.

When placing video inside an element, the video inside that element is positioned and scaled by any positioning and resizing of the element due to style, If the implementation does not support arbitrary scaling of video, then the resource shall be treated as not available and the fallback elements (if any) applied, otherwise the video shall be scaled as required. The video is always scaled to the full size of the element.

NOTE: Since the video is scaled to exactly fit the elements box, the `overflow` and `clip` attributes have no meaning for such elements.

Video element objects are treated as defined by the styling of the element. However, it is not required to support `opacity` style on such elements.

8.8.5.8.1.1 dvb-clip-video

The mapping to the source video is provided by the `dvb-clip-video` property which allows an arbitrary portion of the video to be placed within the element.

dvb-clip-video: The dvb-clip-video property defines the bounding box of the source rectangle in the source video allowing an arbitrary portion of the video to be placed within the element (as in GEM [1] figure 36 "Introducing video into the AWT component stack"). The coordinate space used to express the region is that of the decoded video after possible TR 101 154 [i.1] up-sampling. This rectangular segment of the video is then used to fill the element to which the style property is applied.

EXAMPLE:

```
<object src="dvb://current" style="dvb-clip-video:rect (10pels, 50pels, 70pels, 90pels)"/>.
    Value: <shape>
    Initial: full video rectangle
    Applies to: object, image
    Inherited: no
    Percentages: N/A
    Media: visual
    Media: visual
```

<shape>: See table 13, "MHP profile of CSS data types". In this context <top>, <bottom> <right>, and <left> specify <pels>.

<pels>: The offset is specified in the pels of the input video and indicates an offsets from the left or top edge of the video as appropriate.

8.8.5.9 DOM Access to CSS

In order to gain programmatic access to the `@dvb-viewport` rule the language specific binding for the DOM interface specified in clause 8.11.8.1.3, "DVBCSSViewportRule" may be used.

8.8.5.10 Focus traversal and short-cuts

In MHP it is possible to determine the order of traversal of elements using the navigation keys, and to specify certain keys as short cuts.

The `dvb-tv` media type includes the following additional style properties.

Table 16: Navigation properties

Property	Meaning
nav-up	References the visual element to receive focus on receipt of the VK_UP event
nav-down	References the visual element to receive focus on receipt of the VK_DOWN event
nav-left	References the visual element to receive focus on receipt of the VK_LEFT event
nav-right	References the visual element to receive focus on receipt of the VK_RIGHT event
nav-index	Provides a navigation number for the element. This number must be unique on the page
nav-first	An element with this style will be given focus on first presentation of the page

The values of `nav-up, nav-down.nav-left`, and `nav-right` may be `<integer>`, in which case the referenced element is the element with the corresponding `nav-index` style, or they may be `<string>`. The `<string>` has the following syntax:

```
value:          '[outer]' | [frame-ref] '#' [elem-ref]
frame-ref:      frameid |
elem-ref:       '$'integer | elemid
```

where:

frameid: is the id of the frame containing the referenced item (the current frame is the default). The same rules shall apply as for the target attribute in HTML 4 [41], except that steps 3 and 4 of clause B.8 therein shall not apply as DVB-HTML does not open new windows.

elemid: is the id of the referenced item. At least one of the `frame-ref` or `elem-ref` optional elements must be present if value is not `[outer]`.

integer: is a positive integer, and represents the element with the corresponding `nav-index` style (the default is the element with the `nav-first` style).

e.g.:

```
<style type="text/css">
    #foo nav-right:"frame1#bar"
</style>
```

Would indicate the style for the element with the `id` foo, would navigate the focus to the element with the `id` "bar" in the frame "frame1".

The following string is reserved and has special meaning:

[outer]: This string refers to the external application context in the case that the DVB-HTML application is an inner application. If this name is used in the case where the application is not an inner application it shall be ignored.

If the element to receive the navigation focus cannot be determined by these rules, then the user agents default focus processing should be applied.

NOTE: The navigation hints should be used with some care, as the user agent may have applied defaults based on the current layout. If an author overrides these defaults, then the layout may produce non intuitive results for the viewer.

8.8.6 Font selection

There are four possible font selection actions defined in CSS2: name matching, intelligent matching, synthesis, and download. A user agent shall support the font matching algorithm as defined in section 15.5 in CSS 2 [39]. The user agent is not required to support intelligent font matching algorithm and font synthesis or their associated descriptors as described in table 17, "Descriptor support in MHP". The user agent shall support the CSS2 `@font-face` rule and shall ignore any font index file co-located with a DVB-HTML application.

Table 17: Descriptor support in MHP

Descriptor	Required
ascent	
baseline	
bbox	
cap-height	
centerline	
definition-src	
descent	
font-family	Yes
font-size	Yes
font-stretch	Yes
font-style	Yes
font-variant	Yes
font-weight	Yes
mathline	
panose-1	
slope	
src	Yes
stemh	
stemv	
topline	
unicode-range	Yes
units-per-em	Yes
widths	
x-height	

8.8.6.1 Restrictions on "src" descriptor

The user agent shall fully support the descriptor for Referencing: `src` with the following restrictions:

- The only format type required to be supported for the font `src` descriptor is `"truedoc-pfr"`.
- The only string required to be supported for the <font-face-name> value for access to local fonts is `"Tiresias"`, see GEM [1], clause G.4.1, "The built-in font".

8.8.7 Font specification

The user agent is required to support the following set of properties for selecting a font:

- `font-family`.
- `font-style`.
- `font-variant`.
- `font-weight`.
- `font-stretch`.
- `font-size`.
- `font-size-adjust`.
- `font`.

Setting `font-stretch` and `font-size-adjust` is not required to have any visual effect in a constrained graphics environment.

The text rendering model used by the user agent shall respect the text rendering model defined in GEM [1], clauses D.3.4, D.3.5, D.3.6.1 and D.3.6.2.

8.8.8 Default behaviour

In the absence of any other styling information for a page, the user agent shall render using the default stylesheet in annex AB, "DVB HTML StyleSheet".

8.8.8.1 Default style sheet font rules

For DVB HTML user agents the style sheet referred to by CSS 2 (see UA default stylesheet in CSS 2 [39]) will consist of the stylesheet in annex AB. This stylesheet is not directly accessible to DVB HTML applications.

In addition to the style entries listed in the appendix, this stylesheet shall also provide `@font-face` rules for all installed fonts, which shall at least cover the minimum installed set of fonts (see GEM [1], clause G.4, "Resident fonts and text rendering") and have rules for the CSS 2 generic families, `serif, sans-serif, cursive, fantasy` and `monospace`.

The minimum requirement of the additional font rules are:

- The implementation-specific stylesheet shall provide a font description with name "`Tiresias`".
- The implementation-specific stylesheet shall provide the normal weight.
- The implementation-specific stylesheet shall provide the normal stretch.
- The implementation-specific stylesheet shall provide at least the point sizes required (see GEM [1], table 85, "Example font sizes and use cases").
- The implementation-specific stylesheet shall provide a `src`: to locate the font data.
- The implementation-specific stylesheet shall not provide `font-style` or `font-variant` declarations unless the implementation also provides font data that would visually distinguish the rendered text.

For example a minimal implementation might use the following rule:

```
@font-face {
    font-family: "Tiresias", serif, sans-serif, cursive, fantasy, monospace;
    font-weight: normal;
    font-stretch: normal;
    font-size: 36pt, 31pt, 26pt, 24pt;
  }
```

8.8.8.1.1 Extending the simple rule

An implementation may additionally include `unicode-range`: data for the installed fonts, and other "Descriptors for Matching" (CSS 2 [39], section 15.3.6), which might be used, e.g. for `font-size-adjust`.

8.8.8.1.2 Fallback for italic, small caps and font stretch

The rules above imply that the available fonts cover all variations of `italic`, `small-caps` and `font-stretch`. An implementation that only had the minimum installed fonts should present the same (plain) style glyphs for any combination of the styles required by CSS 2.

NOTE: This allows an author to use `font-style: italic, font-weight: bold` etc. in stylesheets without having text so styled appear as missing character glyphs when using Tiresias, but they should be aware that no extra styling information may be displayed.

8.9 Xlet integration

DVB-HTML defines an `<object>` element that may be used for embedded Xlets. For `<object>` elements used for embedded Xlets the type should be set to "`application/dvbj`", the `<object>` element may contain `<param>` elements containing run time arguments for the Xlet defined by the `<object>` element. The sequence of these arguments is mapped to XletContext.ARGS and shall be available to the Xlet, in the form of an array of Strings, by calling:

```
javax.tv.xlet.XletContext.getXletProperty(XletContext.ARGS)
```

If no valid arguments are defined, this method shall return an array of strings with zero elements.

By default the application identifier of Xlets contained in DVB-HTML are the same as that of the enclosing application.

Using the AppID attribute the embedding HTML context can supply a different AppID for the Xlet, thus giving it a different context (e.g for persistent file locations). The value given for the AppID attribute must be one of those given in the DVB-HTML application descriptor (see clause 10.10.1). An Xlet with an AppID assigned by the HTML application is still considered part of the single DVB HTML application, e.g. for inter application communication.

If an Xlet and an embedded Xlet have the same application identifier it is only possible for one to be running at any time. So, if the Xlet has already started then a DVB-HTML application shall not be able to start an embedded Xlet with the same application identifier. Conversely, if the embedded Xlet has already started the Xlet shall not start. Any lifecycle signalling associated with an Xlet with the same application identifier as an embedded Xlet shall not apply to the embedded Xlet. If a DVB-HTML application attempts to start an embedded Xlet that has an application identifier that conflicts with an already running application:

- the embedded Xlet shall not be started;
- the already running application is not considered apart of the DVB-HTML application.

8.9.1 Object element

When used to include an Xlet inner application the following object element attributes semantics shall apply:

declare: when present, makes the current object definition a declaration only. The object must be instantiated by a subsequent object definition referring to this declaration.

classid: specifies the location of the class that implements the interface javax.tv.xlet.Xlet. Exactly one instance of this Xlet class shall be created. The URI shall be evaluated as relative to the codebase attribute.

codebase: specifies the base URI for the Xlet. Xlet classes are loaded relative to the "directory" specified by this locator. If this attribute is not specified, then it defaults to the same base URI as for the DVB-HTML document carrying the object element. If the URI is not within the application containing the Xlet then the Xlet shall not be loaded.

codetype: specifies the content type of the data referenced by the classid and codebase attributes. In the case of Xlets it will be set to the MIME type for Xlets (application/javatv-xlet).

archive: this is ignored for embedded Xlets.

standby: specifies a message that a user agent may render while loading the objects implementation and data.

height, width: these attributes specify the size of the Xlet's visible representation. If either height or width is zero or negative, then the method javax.tv.graphics.TVContainer.getRootContainer() shall return null.

8.9.2 Param element

The param element is defined for embedding inner application Xlets as follows:

```
<!ELEMENT param EMPTY>
<!ATTLIST param
    id          ID                  #IMPLIED
    name        CDATA               #REQUIRED
    value       CDATA               #IMPLIED
    valuetype   (data|ref|object)   "data"
    type        CDATA               #IMPLIED
>
```

where the specified attributes are defined as follows:

id: is a unique identifier for the param element. The scope of uniqueness is the document carrying the param element.

name: should be set to either arg_n (case-insensitive), where n is a non negative integer, or, appid.

- Where the name is of the form arg_n this specifies that the value argument contains the value of run time argument number n for the Xlet. If the largest value for n among the param tags is x, then the ARGS array shall contain x+1 elements. If two param elements with the same name are given for an Xlet, the corresponding array element shall contain the value of one of them. Which one is chosen is not specified.

  These arguments are available to an Xlet through XletContex.getXletProprty(XletContext.ARGS) as defined in Java TV [22].

- The appid shall carry a 16 bit bit string (left bit first) encoded as follows:

  appid = "0x" 4*hex

  hex = digit | "A" | "B" | "C" | "D" | "E" | "F" | "a" | "b" | "c" | "d" | "e" | "f"

  digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

  If more than one parameter tag with the name appid is provided or the value does not correspond to that of one of the application ids listed in the DVB-HTML application descriptor (see clause 10.10.1) the Xlet shall not be started.

value: is the string value of this run time argument.

valuetype: ignored. Xlet arguments can only be strings, so it does not make sense to provide values of any other type.

type: ignored. This follows from the treatment of valuetype.

8.9.3 Example

The following is an example of using object and param elements in order to include an Xlet inner application in a DVB-HTML document:

```
<object
    id = "slideShow"
    codetype = "application/dvbj"
    classid = "slideShow.class"
    codebase = "first_xlet/"
    height = "180"
    width = "320">
    <param name = "arg_0" value = "Everest"/>
    <param name = "arg_1" value = "Kilimanjaro"/>
    <param name = "appid" value = "0xbded"/>
</object>
```

8.10 Scripting

ECMAScript is a simple lightweight object-oriented scripting language for manipulating computational objects within a host environment. The present document requires ECMAScript as defined in ECMA-262 [33].

8.10.1 DOM 2 binding

The present document requires the DOM 2 binding libraries for the interfaces required in clause 8.11, "Document Object Model (DOM)" as defined in the respective references.

8.10.2 Interface between ECMAScript and DVB-J

By allowing DVB-J to expose APIs to ECMAScript, applications may be created which are a hybrid of DVB-J and DVB-HTML/ECMAScript. For example, the core engine for a particular application could be implemented in DVB-J and the user interface and graphical content implemented in DVB-HTML/ECMAScript.

8.10.2.1 ECMAScript APIs for accessing DVB-J

Public DVB-J: packages, classes, methods and fields shall be visible in ECMAScript using a property of the global object called Packages. For example, RMI's Naming class would be accessed using Packages.java.rmi.Naming. The java.lang package is accessed using Packages.java.lang. In addition to DVB-J classes, class files located in the codebase as specified by the meta tag described below are also accessible.

Xlets may be accessed via the DOM or accessed via the inter application communication API, see clause 11.7.3, "Inter-Application and Inter-Xlet communication API".

When ECMAScript uses the Packages object to reference a Java class that is not loaded, the codebase is searched for the class file. The codebase is specified using a meta element with a name "codebase" and the "content" specifying the classpath as a base URL, e.g.:

```
<meta name="codebase" content="http://example.com/sports/">
```

If multiple codebase meta elements are present in the root document, each specified codebase is searched in its order of appearance. If the class is not found, a java.lang.ClassNotFound exception is thrown. Only meta elements on the root document of the DVB-HTML application may specify the codebase; others have no effect.

ECMAScript contexts within an application share a single classloader. This classloader is separate from that of any Xlets that are part of the DVB HTML application.

8.10.2.2 Inter-Xlet and Xlet-ECMAscript Communication via org.dvb.ixc

In MHP, Xlets are allowed to communicate with each other by exporting objects via the org.dvb.io.ixc APIs described in clause 11.7.3, "Inter-Application and Inter-Xlet communication API" and annex Y, "Inter-application and Inter-Xlet communication API". When an Xlet wishes to export an object so that it is visible to other applications, it is exported under a name formed from a string that contains a unique application identifier, possibly with an arbitrary name appended.

With Xlets embedded in a DVB-HTML document, there is the possibility that there will be multiple active Xlets that are part of the same application. Each such Xlet will have its own classloader (see clause 8.10.2.1), and will not be able to directly share instances with other Xlets. The Xlets will be able to communicate with each other, using the inter-xlet communication mechanism, accessed through the org.dvb.io.ixc APIs.

The MHP naming convention for exported objects are preserved. The APIs provide two options for exporting objects: Exporting them so that they are only visible within the same application, or exporting them so that they are visible to other applications. In either case, it is possible for two Xlets to export an object under the same name. If this happens, the last object exported shall be the one that is visible; an export from one Xlet might therefore overwrite an export from another that occurred earlier in time.

NOTE: Xlets that are in the same DVB-HTML document (and part of the same application) are expected to be authored to work together, so it is the application author's responsibility to ensure that no harmful overwriting occurs.

8.10.2.3 Security

The DVB-J security model applies to the application as a whole, hence an Xlet embedded in a DVB-HTML document has the privileges of the overall DVB-HTML application. ECMAScript may directly invoke DVB-J with the same permissions as the overall application.

See also clause 8.14, "Security of DVB-HTML applications".

NOTE: ECMAScript can pass both internal and external strings across the bridge to Java, but Java has no way to distinguish between them. Hence, Java code written for use across the bridge should treat such strings as it would any other strings from untrusted sources.

8.10.2.4 Implicit Method Selection

When more than one method signature matches the set of arguments, argument preference is compared for each argument starting with the first, until one signature has a preferred match. Preference is determined from the argument conversion tables in the following clause. In those tables, type conversions are listed in decreasing order of preference.

8.10.2.5 Explicit Method Selection

Methods may be selected explicitly by referring to a method as if it were an element of an ECMAScript associative array indexed by the signature, e.g.:

```
new Packages["java.lang.String"]["(char[])"](c);
```

8.10.2.6 Static Method Invocation

Static methods of a Java class shall be invocable from ECMAScript either on the JavaClass object or on an instance of the class.

8.10.2.7 Method Signature Matching

Typically, methods invoked from ECMAScript are designed for that purpose and unambiguous. In most cases, it is probably sufficient to leave method selection up to the implementation, however the tables below define both the preferred order used for selecting signatures and the type conversion invoked when an argument is passed.

Method selection is performed similarly to DVB-J. The DVB-J method must:

- Be public.
- Be static or non-static depending on whether the ECMAScript invocation is on the class or on an object, respectively.
- Have the same number of arguments.
- Have arguments with types to which the ECMAScript arguments can be converted according to the tables below.

If these conditions are not met, the DVBException with the error code CONVERSION_TYPE_ERR is raised.

If there is only one DVB-J method matching these criteria, that method is invoked.

If more than oneDVB-J method matches, then a method is preferred over another for invocation when the conversion of at least one of its arguments is preferred and the conversion of all other arguments is equivalent or preferred. In cases where no method is preferred over all others, results may be implementation dependent.

8.10.2.8 New ECMAScript Object Types

ECMAScript is a loosely typed language with few fundamental types (mainly: Number, Boolean, String, Object, Null and Undefined). In order to access DVB-J 's richer type structure, three new ECMAScript object types are introduced to support values returned from calls to DVB-J.

Table 18: New ECMAScript Object Types

ECMAScript Object Type	Description
JavaArray	Wrapper for a DVB-J array. Behaves like an ECMAScript array including being indexed by an integer and having a length property.
JavaObject	Wrapper for a DVB-J Object. Converting this wrapper to a string, calls the toString() method. Converting to a number calls the doubleValue() method. Converting to a boolean results in true, unless the object is null.
JavaClass	Wrapper for a DVB-J class.

8.10.2.9 Type Conversion (ECMAScript to DVB-J)

In the following tables conversions are listed in decreasing order of preference (from top to bottom within the table and from left to right within each table row). Conversions that are not listed are not allowed.

When the ECMAScript parameter is a boolean, conversion is performed as follows.

Table 19: Conversion from ECMAScript boolean

DVB-J Parameter Type	Conversion from ECMAScript boolean
boolean	Direct conversion.
java.lang.Boolean java.lang.Object	Object of type java.lang.Boolean.
java.lang.String	Converted to "true" or "false".
double, float, long, int, short, char, byte	1, if true. 0, if false.

When the ECMAScript parameter is a number, conversion is performed as follows.

Table 20: Conversion from ECMAScript number

DVB-J Parameter Type	Conversion from ECMAScript Number
double	Direct conversion with full precision.
java.lang.Double	java.lang.Double created with full precision.
Float	Rounded to float precision. Values out of float range become +infinity or -infinity.
long, int, short, char, byte	Rounded to appropriate precision. Values out of range causes the DVBException with the error code CONVERSION_TYPE_ERR to be raised. Passing NaN causes the DVBException with the error code CONVERSION_TYPE_ERR to be raised.
java.lang.String	Conversion to a string using the ECMAScript toString function.
boolean	Passing NaN causes the DVBException with the error code CONVERSION_TYPE_ERR to be raised. 0 is false. Non-zero is true.
java.lang.Object	java.lang.Double created with full precision.

When the ECMAScript parameter is a string, conversion is performed as follows.

Table 21: Conversion from ECMAScript string

DVB-J Parameter Type	Conversion from ECMAScript string
java.lang.String java.lang.Object	Object of type java.lang.String.
char	If string is a single character, conversion via the ECMAScript charCodeAt() function; otherwise the DVBException with the error code CONVERSION_TYPE_ERR is raised.

When the ECMAScript parameter is undefined, conversion is performed as follows.

Table 22: Conversion from ECMAScript undefined

DVB-J Parameter Type	Conversion from ECMAScript undefined
java.lang.String java.lang.Object	Object of type java.lang.String with a value of "undefined".
boolean	False
long, int, short, byte, char	Zero
double, float	Nan

When the ECMAScript parameter is null, conversion is performed as follows.

Table 23: Conversion from ECMAScript null

DVB-J Parameter Type	Conversion from ECMAScript null
Any object	Null
boolean	False
double, float, long, int, short, byte, char	Zero

When the ECMAScript parameter is JavaArray, JavaObject or JavaClass, conversion is performed as follows.

Table 24: Conversion from ECMAScript DVB-J objects

DVB-J Parameter Type	Conversion from ECMAScript DVB-J Objects
Most specific assignment compatible class	Unwrap ECMAScript JavaObject to original DVB-J Object. ECMAScript JavaObject is of an incompatible class to the required class the DVBException with the error code CONVERSION_TYPE_ERR shall be raised.
java.lang.String	Result of `toString()`.
double, float, long, int, short, char, byte	Result of `doubleValue()` is taken and converted as a Number to the DVB-J type as above. If no `doubleValue()` method is defined, causes the DVBException with the error code CONVERSION_TYPE_ERR to be raised.

When the ECMAScript parameter is an ECMAScript Object, conversion is performed as follows.

Table 25: Conversion from ECMAScript Object

DVB-J Parameter Type	Conversion from ECMAScript Object
java.lang.Object (including subclasses) or interface	If ECMAScript object has all public fields and methods of the DVB-J parameter type, and has been generated as specified in Subclassing and Interface Instance Creation, then the generated DVB-J class is used. Otherwise the DVBException with the error code CONVERSION_TYPE_ERR shall be raised.

ECMAScript Arrays are passed as parameters only to Java methods accepting an array as its corresponding argument. An ECMAScript array containing a single element type matches a Java array parameter based on the matching preference of the array element types. An ECMAScript array containing different element types is not convertible to and does not match any Java parameter type.

8.10.2.10 Subclassing and Interface Instance Creation

Instances of an instantiable Java class can be create by using the "new" operator on the Java class, e.g.:

```
    new Packages.java.lang.String("astring");
```

The parameters are passed to the constructor using the type conversion rules above. This mechanism is extended to allow the creation of instances of classes that must be subclassed to produce an instantiable class. The SubType object is defined and used with a constructor function to allow ECMAScript to construct an instance of a subclass. The Subtype constructor takes a string naming the Java class to be subclassed as its argument, e.g. "java.awt.ActionListener" and an ECMAScript function which acts as a constructor to initialize any abstract elements. Invoking the Subtype constructor on a final Java class shall cause the DVBException with the error code SUBCLASS_NOT_ALLOWED_ERR to be raised. The object returned by the Subtype constructor is a function used to create instances. When this function is used as a constructor, its arguments are passed to the function specified in the Subtype constructor. The function is invoked with the current object being an ECMAScript handle to the JavaObject being instantiated.

EXAMPLE:

```
function actDone(event) {
    /* My event handling method */
}

function MyListenerConstructor(arg1, arg2) {
    /* if ActionListener had a constructor it could be called here
    * e.g. this.ActionListener(arg2);
    */
    this.ActionPerformed = arg1;
}
MyListenerType = new Subtype("java.awt.ActionListener", MyListenerConstructor);
MyListener = new MyListenerType(actDone, 0);

NOTE:   This mechanism may be used to create listeners from ECMAScript.
An example of constructing and setting a listener:
<script>
    function receiveEvent(x) {
        window.open("http://chat.tv.com/chatLogin?user="+x, _chatFrame);
    }
    function chatConstructor() {
        this.receiveUserPreferenceChangeEvent = receiveEvent;
    }
    chatListenerType = new Subtype("org.dvb.user.UserPreferenceChangeEvent", chatConstructor);
    chatListener = new chatListenerType();

    Packages.org.dvb.user.UserPreferences.addUserPreferenceChangeListener(chatListener);
</script>
```

8.10.2.11 Type Conversion (DVB-J to ECMAScript)

Type conversion of return values occurs as follows.

Table 26: Conversion from DVB-J types

DVB-J Return Value Type	Conversion to ECMAScript type
double, float, long, short, int, byte, char	Direct conversion to number
boolean	Direct conversion to boolean
array	JavaArray
java.lang.Class	JavaClass
Object wrapped around ECMAScript Object	Direct conversion (unwrapping)
Other objects (including java.lang.Double, java.lang.Integer and java.lang.String (see note), java.lang.Exception)	JavaObject (an ECMAScript wrapper around the DVB-J Object providing the same methods
NOTE: The explicit or implicit invocation of the ECMAScript toString() function on the ECMAScript handle to a java.lang.String shall return an external string.	

8.10.2.12 Catching DVB-J Exceptions in ECMAScript

DVB-J exceptions are thrown back into ECMAScript when the underlying DVB-J method throws an exception, or an exception is generated as part of the attempted type conversion from ECMAScript to DVB-J (see type conversions above). For example, the following ECMAScript:

```
try {
        Packages.java.lang.Class.forName("someClass");
} catch (e) {
        print("Exception: " + e);
}
```

would print out something like the following, if someClass does not exist.

```
Exception: java.lang.ClassNotFoundException: someClass.
```

8.11 Document Object Model (DOM)

DOM is a platform and language neutral interface that provides programmatic access to the content, structure and style of documents. Support for language bindings for the DOM for both Java language and ECMAScript is required.

Table 27: Supported DOM modules

DOM Module		Where specified	Required
Package	**Feature String**		
Level 2 Core	Core	Document Object Model (DOM) Level 2 Core Specification [34]	Yes
	XML		No
Level 2 Views	Views	Document Object Model (DOM) Level 2 Views Specification [38]	Yes
Level 2 Stylesheets	StyleSheets	Document Object Model (DOM) Level 2 Style Specification [37]	No
Level 2 CSS stylesheets	CSS	Document Object Model (DOM) Level 2 Style Specification [37]	No
	CSS2		Yes
Level 2 Events	Events	Document Object Model (DOM) Level 2 Events Specification [36]	Yes
	UIEvents		Yes
	MutationEvents		Yes
DVB-HTML	DVBHTML	See clause 8.11.4, "DVB-HTML DOM module"	Yes
DVB Events	DVBEvents	See clause 8.11.2, "DVB Events DOM module"	Yes
DVB Key Events	DVBKeyEvents	See clause 8.11.3, "DVB Key events DOM module"	Yes
DVB CSS	DVBCSS	See clause 8.11.8.1, "DVB CSS DOM module"	Yes
DVB Environment	DVBEnvironment	See clause 8.11.7, "DVB Environment object module"	Yes

8.11.1 DOM Level 2 Events

8.11.1.1 Fundamental interfaces

DVB-HTML shall support the events module as defined in Document Object Model (DOM) Level 2 Events Specification [36] (which includes the following interfaces: EventTarget, EventListener, DocumentEvent, Event and EventException).

8.11.1.2 Event interfaces

The following event sets from Document Object Model (DOM) Level 2 Events Specification [36] shall be supported:

- UIEvent.
- MutationEvent.

It is not required to support the following event sets:

- HTML events.

NOTE: Equivalents to the Load and Unload events are provided, see clause 9.3.4, "Application activity events".

- MouseEvent.

  It is recommended that implementations providing support for mouse like devices provide the MouseEvent event set.

8.11.2 DVB Events DOM module

A DOM application can use the hasFeature method of the DOM implementation interface to determine that this module is supported. The feature string for all interfaces listed in this module is "DVBEvents" and the version "2.0".

8.11.2.1 Key events

DVB-HTML will provide the W3C event bindings for keyboard and remote control events in a later release of the present document when the W3C Dom level 3 events specification is finalized. Until such time, scripting will have access to key events as specified in an additional DVB DOM module specified in clause 8.11.3, "DVB Key events DOM module".

8.11.2.2 Lifecycle events

This clause provides the mapping of DVB-HTML life cycle events to the DOM event model.

8.11.2.2.1 Interface DVBLifecycleEvent

The DVBLifecycleEvent interface provides specific contextual information to a DVB-HTML application associated with its lifecycle.

All lifecycle events are delivered to the root element of the document in the root frame of an application and follow the normal event delivery rules in Document Object Model (DOM) Level 2 Events Specification [36]. For clarification, sub-frames which contain the root of separate applications (as described in clause 8.14.3, "Inter application security") also have DVB lifecycle events delivered to the corresponding document root. The different types of such events that can occur are detailed under clause 8.11.2.2.2, "Event definitions".

8.11.2.2.1.1 IDL Definition

```
// Introduced in DVB-HTML:
interface DVBLifecycleEvent : Event {
    readonly attribute DOMString eventinfo;
    void initDVBLifecycle Event(in DOMString typeArg,
        in boolean canBubbleArg,
        in boolean cancelableArg,
        in long detailArg);
};
```

8.11.2.2.1.2 Attributes

eventinfo: This attribute specifies additional information about the event the meaning of which depends on the type of the event.

8.11.2.2.1.3 Methods

Table 28: DVB Lifecycle Event Method

Method	Name	Description		
	initDVBLifecycle Event	This method is used to initialize the value of a lifecycle event created through the `Document_Event` interface. This method may only be called before the lifecycle event has been dispatched via the `dispatchEvent` method, though it may be called multiple times during that phase if necessary. If called multiple times, the final invocation takes precedence.		
Parameters	**Name**	**Type**	**Qualifier**	**Description**
	typeArg	DOMString		Specifies the event type. The string is identical to the name of the event (e.g. the AppStarting event is specified with the string "AppStarting").
	canBubbleArg	Boolean		Specifies whether or not the event can bubble.
	cancelableArg			Specifies whether or not the event's default action can be prevented.
	detailArg	Number		Specifies the Event's detail.
Return value	**Type**	**Description**		
No Return Value.				
Exceptions				
No Exceptions.				

8.11.2.2.2 Event definitions

8.11.2.2.2.1 AppStarting

This event is given to the DVB-HTML application whilst in the Active state if and only if it is signalled as PREFETCH. The user agent sends this event when it receives the dvb.start event. After receiving the dvb.start event and prior to sending this event the user agent begins displaying the DVB-HTML application.

- Bubbles: No.
- Cancelable: No.
- Context Info: None.

8.11.2.2.2.2 AppActive

This event is given to the DVB-HTML application on transitioning from the Loading state to the Active state.

- Bubbles: No.
- Cancelable: No.
- Context Info: eventinfo.

The values of eventinfo are those defined in table 68, "DVB-HTML application control code values" with the following semantics:

- PREFETCH means that the DVB-HTML application will not be presented by the user agent until the user agent has sent an AppStarting event.
- All other values imply that the DVB-HTML application will not receive an AppStarting event and can assume that presentation has begun.

8.11.2.2.2.3 AppPause

This event is given to the DVB-HTML application on transitioning to the Paused state.

- Bubbles: No.
- Cancelable: No.
- Context Info: None.

8.11.2.2.2.4 AppResume

This event is given to the DVB-HTML application on transitioning out of the Paused state to the Active state.

- Bubbles: No.
- Cancelable: No.
- Context Info: None.

8.11.2.2.2.5 AppDestroyed

This event is given to the DVB-HTML application on transitioning to the Destroyed state.

- Bubbles: No.
- Cancelable: No.
- Context Info: None.

8.11.2.2.2.6 AppKilled

This event is given to the DVB-HTML application on transitioning to the Killed state.

- Bubbles: No.
- Cancelable: No.
- Context Info: None.

8.11.2.2.2.7 AppTerminating

This event is given to a DVB-HTML application when an application within a subframe terminates.

- Bubbles: No.
- Cancelable: No.
- Context Info: eventinfo.

The value of eventinfo is the name of the frame where the terminating application was hosted.

8.11.2.2.3 State transition summary

The following table summarizes the allowed transitions in the lifecycle model (see clause 9.3.3, "The State Model").

Table 28a

State transition	Loading	Active	Paused	Destroyed	Killed
Loading to	no	yes 1	no	yes 2	yes 3
Active to	no	no	yes 4	yes 5	yes 6
Paused to	no	yes 7	no	yes 8	yes 9
Destroyed to	no	no	no	no	yes 10
Killed to	no	no	no	no	no

The following table describes the events delivered to the application following these transitions.

Table 28b

Cross reference	Transition	Event delivered
1	Loading to Active	event AppActive on entry to Active state
2	Loading to Destroyed	event AppDestroyed on entry to Destroyed state
3	Loading to Killed	event AppKilled on entry to Killed state
4	Active to Paused	event AppPause on exit from Active state
5	Active to Destroyed	event AppDestroyed on entry to Destroyed state
6	Active to Killed	event AppKilled on entry to Killed state
7	Paused to Active	event AppResume on entry to Active state
8	Paused to Destroyed	event AppDestroyed on entry to Destroyed state
9	Paused to Killed	event AppKilled on entry to Killed state
10	Destroyed to Killed	event AppKilled on entry to Killed state
NOTE: In 8 and 9 no AppResume event be received.		

8.11.2.3 Additional DVB Events

8.11.2.3.1 Trigger events

See clause 8.7, "Synchronization".

8.11.2.3.2 DVBDOMStable event

This event signals the completion of the current document DOM construction. After the receipt of this event the DOM structure of the document will be complete, however the DOM structure in sub-frames or in-line frames may not be complete and resources such as the contents of images may not have finished loading. Prior to the generating the event implementations are free to have an incomplete DOM structure referenced by the document variable.

This event uses the base Event interface to pass contextual information. The following type is allowed:

DVBDOMStable: This event occurs when the DOM implementation is fully constructed, but is not required to wait until all of the associated frames in a `<frameset>`, content in `<image>` or `<object>` elements loads. This event is valid for `<body>` and `<frame>` elements.

- Bubbles: No.

- Cancelable: No.
- Context Info: None.

8.11.2.3.3 DVB-HTML events

These events use the base Event interface to pass contextual information. The following types are allowed:

load: The load event occurs when the DOM implementation finishes loading all content within a document, all frames within a `<frame>`, or an `<object>` element.

- Bubbles: No.
- Cancelable: No.
- Context Info: None.

unload: The unload event occurs when the DOM implementation removes a document from a window or frame. This event is valid for `<body>` and `<frame>` elements.

- Bubbles: No.
- Cancelable: No.
- Context Info: None.

8.11.3 DVB Key events DOM module

A DOM application may use the hasFeature(feature, version) method of the DOMImplementation interface with parameter values "DVBKeyEvents" and "2.0" (respectively) to determine whether or not the event module is supported by the implementation. The feature string is also used with the createEvent method. It is implementation dependent how a MHP gets the set of supported keys.

Key events generated by a real or virtual keyboard or from a remote control device shall be generated by the user agent and can be registered for by applications through the DOM. Registering for events using the type string "HKeyEvent" or "HRcEvent" can place a listener that shall receive an object that implements the DVBKeyEvent interface. All Key events are initially targeted at the element with focus, or the body element if no element has focus.

e.g.:

```
function myListener(evt) { ... }
node.addEventListener("HRcEvent", myListener, false )
```

8.11.3.1 Interface DVBKeyEvent

The DVBKeyEvent interface provides specific contextual information associated with Remote Control and Keyboard Events. The interface is able to describe the representation of a key event as a string, color or symbol (such as a triangle, ">", for "play"). This allows an application to describe a button on an input device correctly for a given platform.

The keyChar attribute describes the representation of a key event as a string. All available events shall have a text representation. Other representations may be available.

NOTE: The DVBKeyEvent interface is designed to be compatible with the W3C level 3 DOM key events albeit with some redundancy, and in future versions of the specification it is envisaged that objects will also support the W3C defined interface.

8.11.3.1.1 IDL Definition

```
// Introduced in MHP 1.1:
interface DVBKeyEvent : UIEvent {
readonly attribute unsigned long when;
readonly attribute unsigned long modifiers;
readonly attribute unsigned long keyCode;
readonly attribute DOMString keyChar;
readonly attribute DOMString color;
readonly attribute DOMString symbol;
    void initDVBKeyEvent(in DOMString type,
        in boolean canBubbleArg,
        in boolean cancelableArg,
        in unsigned long modifiers,
        in unsigned long keycode,
);
```

8.11.3.1.2 Attributes

Table 29: Key event attributes

Name	Readonly	Type	Description
when	yes	unsigned long	The timestamp for the event.
modifiers	yes	unsigned long	Indication of any modification keys active for the event.
keyCode	yes	unsigned long	The key code of the key associates with this event.
keyChar	yes	unsigned long	The character representation of the key associated with this event.
color	yes	DOMString	The six coloured key events (VK_COLORED_KEY_0, , VK_COLORED_KEY_5), if implemented, must also be represented by a color. For all other key events this value should be null. The format of the color is specified in CSS 2 [39] section 4.3.6, "Colors".
symbol	yes	DOMString	The URL to an image provided by a MHP to describe the symbol associated with a certain key. The recommended symbols for some Key events are described in HAVi [21]. The value of this attribute can be NULL.

8.11.3.1.3 Methods

Table 30: Key Event Method

Method	Name	Description		
	initDVBKeyEvent	This method is used to initialize the value of a keyboard event. This method may only be called before the key event has been dispatched via the dispatchEvent method, though it may be called multiple times during that phase if necessary. If called multiple times, the final invocation takes precedence.		
Parameters	**Name**	**Type**	**Qualifier**	**Description**
	type	DOMString	-	Either "HRcEvent" or "HKeyEvent".
	canBubbleArg	boolean		Specifies whether or not the event can bubble.
	cancelableArg	boolean		Specifies whether or not the event's default action can be prevented.
	modifiers	unsigned long		The modifiers to be applied to the event.
	keycode	unsigned long	-	The Keycode of the event to be generated.
Return value	**Type**	**Description**		
	-	Initializes the event, all of the attributes shall be set as if this event were generated by the system.		

8.11.4 DVB-HTML DOM module

8.11.4.1 Conformance

A DVB-HTML user agent shall be considered as being an HTML-only DOM implementation as defined in the W3C Document Object Model (DOM) Level 2 Core Specification [34], so that conformance statements in DOM Level 2 applying to those kinds of implementation will apply also to a DVB-HTML user agent. E.g. a DVB-HTML DOM implementation is not required to implement the `createDocument()` method in the `DOMImplementation` interface from the DOM Level 2 Core.

8.11.4.2 Differences from W3C DOM Level 1 HTML interfaces

The W3C DOM 1 HTML interfaces were designed in order to provide convenience functions to manipulate HTML 4.0 documents and also to create documents from scratch. The DOM DVB-HTML module has been defined with the objective to manipulate DVB-HTML documents without any intention to create them. In general, the DOM DVB-HTML module follows the design of the W3C DOM1 HTML interfaces, however the following differences can be observed:

- Some attributes have been made immutable in the DOM DVB-HTML module. These modifications have been indicated in the present document.
- Some attributes and interfaces have not been included in the DOM DVB-HTML module.
- For the length attribute in all interfaces, the type has been homogenized to unsigned long.
- Some semantics have been added to clarify user agent behaviour (e.g. on modification of the disabled attribute).
- Some semantics have been altered to take into account the DVB-HTML DTD (e.g.since the `name` attribute is not supported for identification purposes some attribute semantics had to be modified accordingly).

8.11.4.3 Extensions

8.11.4.3.1 Enumerations

In DVB-HTML, some element attributes can be set to a limited number of values. In order to be as much as possible in line with the IDL defined in the W3C DOM 1 HTML, the present document does not use the IDL `enum`. For attributes that take a limited set of values (i.e. an enumeration) the following semantics shall apply:

- If a mutable attribute is set with a value outside of the possible values the DOMException with the error code SYNTAX_ERR is raised and the original value shall be kept.

The bindings reflect this choice, see annexes AC, "ECMAScript Binding" and AD, "Support for DVB-HTML".

8.11.4.3.2 Initial and current values of form controls

Form controls that accept a user input, e.g. text area, option, text, password, checkbox, radio button, have both an initial value and a current value. The names and the types of these values depend on the type of the control. When a control is created, the means by which the initial value is set depends on the control. See the corresponding interface definition clause 8.11.4.6.6, "DVBHTMLFormElement Interface".

When a control is created, the current value is set to the initial value.

The initial value can only be modified by a script. The current value can be modified either by a user interaction or a scripting instruction. If the current value is modified, the display shall be updated to reflect the value changed.

When a form is reset, either by user interaction or by scripting instruction, for each of the controls accepting user input and contained in the form, the current value shall be set to the initial value. The initial value may have been modified between the creation of the control and the reset of the form. As such, the initial value used shall be the value at the time of the reset.

Form controls that do not accept user input, e.g. hidden controls, submit buttons, reset buttons, buttons, submit images, have only one value, a current value. When a control is created, the means by which the current value is set depends on the control. See clause 8.11.4.6.6, "DVBHTMLFormElement Interface". The current value can only be modified by a scripting instruction. In such a case, it is not required to affect the display.

The file input control has a different behaviour from those described above. It accepts a user input but only has a current value. The semantics of the current value of form controls that do not accept user input applies.

8.11.4.4 System aspects

8.11.4.4.1 Access to the document

8.11.4.4.1.1 Java

The DVB DOM Implementation shall implement the org.dvb.dom API as defined in clause 11.13.1, "Document object model APIs". This will provide applications a handle to the DVB-HTML document object model.

8.11.4.4.1.2 ECMAScript

The DVB DOM implementation shall provide to applications a `document` object that represents the current DVB-HTML document.

8.11.4.4.2 DOM DVB-HTML module

A DOM application can use the `hasFeature` method of the `DOMImplementation` interface to determine whether this module is supported or not. The feature string for all the interfaces listed in this module is "DVBHTML" and the version is ".0".

8.11.4.4.3 DOM modification

8.11.4.4.3.1 DOM access

Any DOM call from the DOM DVB-HTML module shall be synchronous. As such, any subsequent calls can assume that they can operate on a DOM structure updated accordingly to the previous call.

8.11.4.4.3.2 Synchronization with renderer actions

Wherever applicable, any required impact on the visual rendering following a DOM call has been described. However, the present document is voluntarily silent on the nature of the relationships between a DOM modification and the renderer, in particular on the synchronization aspects.

8.11.4.4.3.3 Modifications from the DOM Core

A NO_MODIFICATION_ALLOWED_ERR exception shall be raised when attempting to modify an attribute from the DOM Core which equivalent attribute in the DOM DVB-HTML module is defined to be readonly, e.g. this exception shall be raised when modifying the `type` attribute in the `button` element through the DOM Core.

8.11.4.5 Miscellaneous interfaces

This clause describes various utility interfaces of the DOM DVB-HTML module.

8.11.4.5.1 DVB-HTMLCollection Interface

A `DVBHTMLCollection` is a list of DVB-HTML elements. An individual element may be accessed by either ordinal index or the element's `id` attribute.

NOTE: Collections in the DVB-HTML DOM are assumed to be dynamic meaning that they are automatically updated when the underlying document is modified.

8.11.4.5.1.1 IDL Definition

```
interface DVBHTMLCollection {
    readonly attribute unsigned long length;
    Node namedItem(in DOMString name);
    Node item(in unsigned long index);
};
```

The semantics of the attributes and methods associated with this interface are the same as the semantics associated with the `HTMLCollection` interface defined in Document Object Model (DOM) Level 1 Specification [35].

8.11.4.5.2 DVBHTMLDocument Interface

A `DVBHTMLDocument` is the root of the DVB-HTML document hierarchy and holds the entire content. Besides providing access to the hierarchy, it also provides some convenience methods for accessing certain sets of information from the document.

8.11.4.5.2.1 IDL Definition

```
interface DVBHTMLDocument : Document {
    readonly attribute DVBHTMLCollectionanchors;// DVB-HTML Type coherence
    attribute DVBHTMLElementbody;// DVB-HTML Type coherence
    attribute DOMString cookie;
    readonly attribute DOMString domain;
    readonly attribute DVBHTMLCollection forms; // DVB-HTML Type coherence
    readonly attribute DVBHTMLCollection images; // DVB-HTML Type coherence
    readonly attribute DVBHTMLCollection links; // DVB-HTML Type coherence
    readonly attribute DOMString referrer;
    attribute DOMString title;
    readonly attribute DOMString URL;

    void open();
    void close();
    void write(in DOMString text);
    void writeln(in DOMString text);
};
```

The semantics of the attributes and methods associated with this interface are the same as the semantics associated with the `HTMLDocument` interface defined in Document Object Model (DOM) Level 1 Specification [35], with the extensions, restrictions or amendments defined in clause 8.11.4.5.2.2.

8.11.4.5.2.2 Attributes

Table 31: DVBHTMLDocument attributes

Name	Readonly	Type	Description
anchors	yes	DVBHTMLCollection	As defined in Document Object Model (DOM) Level 1 Specification [35]. Only the return type differs.
body	no	DVBHTMLElement	As defined in Document Object Model (DOM) Level 1 Specification [35]. Only the return type differs.
domain	yes	DOMString	See domain in 8.16.6 "Domain".
forms	yes	DVBHTMLCollection	As defined in Document Object Model (DOM) Level 1 Specification [35]. Only the return type differs.
images	yes	DVBHTMLCollection	As defined in Document Object Model (DOM) Level 1 Specification [35]. Only the return type differs.
links	yes	DVBHTMLCollection	As defined in Document Object Model (DOM) Level 1 Specification [35]. Only the return type differs.
title	no	DOMString	As defined in Document Object Model (DOM) Level 1 Specification [35]. In addition, if the user agent makes this visible, then it shall update the rendering upon modification. This attribute is linked to the <title> element changes to one shall be reflected in the other.

8.11.4.5.2.3 Methods

Table 32: DVBHTMLDocument open method

Method	Name	Description		
	open()	As defined in Document Object Model (DOM) Level 1 Specification [35] opens the document for writing. This creates a new empty document in the target to which subsequent writes are made. The transition and the rendering of the new document shall not be performed until a call to the close() method. The ECMAScript context which opened the new document cannot be destroyed until it calls document.close() (see note).		
Parameters	**Name**	**Type**	**Qualifier**	**Description**
Return value	**Type**	**Description**		
	-	This method does not return a value.		
NOTE: document.open() shall not change the URL of the document.				

Table 33: DVBHTMLDocument close method

Method	Name	Description		
.	close()	As defined in Document Object Model (DOM) Level 1 Specification [35]. Rendering of the document is performed as if a link had been followed with that document as a target.		
Parameters	**Name**	**Type**	**Qualifier**	**Description**
Return value	**Type**	**Description**		
	-	This method does not return a value.		

Table 34: DVBHTMLDocument write method

Method	Name	Description		
	write()	As defined in Document Object Model (DOM) Level 1 Specification [35]. An inline non-deferred script performing document.write() feeds additional strings to the parser. The text is parsed into the document's structure model. These strings are sent to the parser sequenced immediately after the closing script tag of the inline script. See also ("Implicit document.open()").		
Parameters	**Name**	**Type**	**Qualifier**	**Description**
Return value	**Type**	**Description**		
	-	This method does not return a value.		

Table 35: DVBHTMLDocument writeln method

Method	Name	Description		
	writeln()	As defined in Document Object Model (DOM) Level 1 Specification [35]. An inline non-deferred script performing document.writeln() feeds additional strings to the parser. The text is parsed into the document's structure model. These strings are sent to the parser sequenced immediately after the closing script tag of the inline script. See also ("Implicit document.open()").		
Parameters	**Name**	**Type**	**Qualifier**	**Description**
Return value	**Type**	**Description**		
	-	This method does not return a value.		

Implicit document.open()

If document.write() is called from a deferred script context and the document has not been explicitly opened for writing, the first write() performs an implicit open of the containing document.

Implicit document.close()

If the deferred inline script context which performed an open (explicit or implicit) exits without calling close on the document, the document is closed implicitly.

When a document is trying to access the value of the `forms`, `anchors`, `links` or `images` attributes before the parsing of the document is complete, the returned value shall be the collection of matching elements at the time of the call.

8.11.4.6 DVB-HTML element related interfaces

8.11.4.6.1 DVBHTMLElement Interface

All DVB-HTML element interfaces derive from this class.

8.11.4.6.1.1 IDL Definition

```
interface DVBHTMLElement : Element {
    attribute DOMString className;
    attribute DOMString dir;
    attribute DOMString id;
    attribute DOMString lang;
    attribute DOMString title;
};
```

8.11.4.6.1.2 Attributes

The semantics of the attributes associated with this interface are the same as the semantics associated with the HTMLElement interface defined in Document Object Model (DOM) Level 1 Specification [35], with the extensions, restrictions or amendments defined in table 36.

Table 36

Name	Readonly	Type	Description
dir	no	DOMString	As defined in Document Object Model (DOM) Level 1 Specification [35]. Trying to set this attribute to a value other than the values allowed in HTML 4 [41] shall cause the DOMException with the error code SYNTAX_ERR to be raised.
lang	no	DOMString	As defined in Document Object Model (DOM) Level 1 Specification [35]. Trying to set this attribute to a value other than the values allowed in HTML 4 [41] shall cause the DOMException with the error code SYNTAX_ERR to be raised.

8.11.4.6.2 DVBHTMLAnchorElement Interface

This interface models the anchor element, corresponding to the DVB-HTML `<a>` tag. It represents a source anchor, the `<A>` tag with the `href` attribute, or a destination anchor, the `<a>` tag with the `id` attribute.

8.11.4.6.2.1 IDL Definition

```
interface DVBHTMLAnchorElement : DVBHTMLElement {
    attribute DOMString accessKey;
    attribute DOMString charset;
    attribute DOMString href;
    attribute DOMString hreflang;
    attribute DOMString target; // Diverge from W3C DOM HTML
    attribute DOMString type;

    void blur();
    void focus();
};
```

The semantics of the attributes and methods associated with this interface are the same as the semantics associated with the `HTMLAnchorElement` interface defined in Document Object Model (DOM) Level 1 Specification [35], with the extensions, restrictions or amendments defined in clause 8.11.4.6.2.2.

8.11.4.6.2.2 Attributes

Table 37

Name	Readonly	Type	Description
target	no	DOMString	The id of the frame to render the resource designated by the URI in. The semantics of the target attribute in HTML 4 [41] apply with the following modification: all references to the name attribute shall be seen as being references to the id attribute.

8.11.4.6.3 DVBHTMLMapElement Interface

This interface models a client-side image map. See the MAP element definition in HTML 4 [41].

8.11.4.6.3.1 IDL definition

```
interface DVBHTMLMapElement : DVBHTMLElement {
    readonly attribute DVBHTMLCollection areas; // DVB-HTML Type coherence
};
```

The semantics of the attributes associated with this interface are the same as the semantics associated with the `HTMLMapElement` interface defined in Document Object Model (DOM) Level 1 Specification [35], with the extensions, restrictions or amendments defined in clause 8.11.4.6.3.2.

8.11.4.6.3.2 Attributes

Table 38

Name	Readonly	Type	Description
areas	yes	DVBHTMLCollection	As defined in Document Object Model (DOM) Level 1 Specification [35]. Its type has been changed to DVBHTMLCollection.

NOTE: The `name` attribute in the map element is deprecated in DVB-HTML. As such, the map is identified only through the `id` attribute.

When a document is trying to access the value of the `areas` attribute before the parsing of the contents of the `map` element is complete, the returned value shall be the collection of matching elements at the time of the call.

8.11.4.6.4 DVBHTMLAreaElement Interface

This interface models an image map area which can triggers an action when the user clicks on it.

8.11.4.6.4.1 IDL Definition

```
interface DVBHTMLAreaElement : DVBHTMLElement {
    attribute DOMString accessKey;
    attribute DOMString alt;
    attribute DOMString href;
    attribute boolean nohref;
    attribute DOMString target; // Diverge from W3C DOM HTML
};
```

The semantics of the attributes associated with this interface are the same as the semantics associated with the `HTMLAreaElement` interface defined in Document Object Model (DOM) Level 1 Specification [35], with the extensions, restrictions or amendments defined in clause 8.11.4.6.4.2.

8.11.4.6.4.2 Attributes

Table 39

Name	Readonly	Type	Description
target	no	DOMString	The id of the frame to render the resource designated by the URI in. The semantics of the target attribute in HTML 4 [41] apply with the following modification : all references to the name attribute shall be seen as being references to the id attribute.

8.11.4.6.5 DVBHTMLButtonElement Interface

`DVBHTMLButtonElement` represents the DVB-HTML `<button>` tag.

8.11.4.6.5.1 IDL Definition

```
interface DVBHTMLButtonElement : DVBHTMLElement {
    attribute DOMString accessKey;
    attribute boolean disabled;
    readonly attribute DVBHTMLFormElement form; // DVB-HTML Type coherence
    attribute DOMString name;
    readonly attribute DOMString type;
    attribute DOMString value;

    void blur(); // Introduced in DVB-HTML DOM
    void focus(); // Introduced in DVB-HTML DOM
};
```

The semantics of the attributes and methods associated with this interface are the same as the semantics associated with the `HTMLButtonElement` interface defined in Document Object Model (DOM) Level 1 Specification [35], with the extensions, restrictions or amendments defined in clauses 8.11.4.6.5.2 and 8.11.4.6.5.3.

8.11.4.6.5.2 Attributes

Table 40

Name	Readonly	Type	Description
form	yes	DVBHTMLFormElement	As defined in Document Object Model (DOM) Level 1 Specification [35]. Only the return type differs.
value	no	DOMString	The button's current value. It corresponds to the value attribute of the DVB-HTML `<button>` tag but shall be interpreted as a current value (as defined in clause 8.11.4.3.2, "Initial and current values of form controls").

8.11.4.6.5.3 Methods

Table 41

Method	Name	Description		
	blur	Removes focus from this element.		
Parameters	**Name**	**Type**	**Qualifier**	**Description**
	-	-	-	-
Return value	**Type**	**Description**		
	-	-		

Table 42

Method	Name	Description		
	focus	Gives focus to this element.		
Parameters	**Name**	**Type**	**Qualifier**	**Description**
	-	-	-	-
Return value	**Type**	**Description**		
	-	-		

8.11.4.6.6 DVBHTMLFormElement Interface

The `DVBHTMLFORMElement` encompasses behaviour similar to a collection and an element. It provides direct access to the contained input elements as well as the attributes of the form element. See the `FORMElement` definition in HTML 4 [41].

8.11.4.6.6.1 IDL Definition

```
interface DVBHTMLFormElement : DVBHTMLElement {
    attribute DOMString action;
    attribute DOMString acceptCharset;
    attribute DOMString enctype;
    readonly attribute DVBHTMLCollection elements; // DVB-HTML Type coherence
    readonly attribute unsigned longlength; // Lengths type coherence
    attribute DOMString method;
    attribute DOMString target; // Diverge from W3C DOM HTML

    void reset();
    void submit();
}
```

The semantics of the attributes and methods associated with this interface are the same as the semantics associated with the `HTMLFormElement` interface defined in Document Object Model (DOM) Level 1 Specification [35], with the extensions, restrictions or amendments defined in clause 8.11.4.6.6.2.

8.11.4.6.6.2 Attributes

Table 43

Name	Readonly	Type	Description
elements	yes	DVBHTMLCollection	As defined in Document Object Model (DOM) Level 1 Specification [35]. Only the type differs.
length	yes	unsigned long	As defined in Document Object Model (DOM) Level 1 Specification [35]. Only the type differs.
method	no	DOMString	As defined in Document Object Model (DOM) Level 1 Specification [35]. Trying to set this attribute to a value other than the values allowed in HTML 4 [41] shall cause the DOMException with the error code SYNTAX_ERR to be raised.
target	no	DOMString	The id of the frame to render the resource designated by the URI in. The semantics of the target attribute in HTML 4.01 apply with the following modification: all references to the name attribute shall be seen as being references to the id attribute.

When a document is trying to access the value of the `elements` attribute before the parsing of the contents of the `form` element is complete, the returned value shall be the collection of matching elements at the time of the call.

8.11.4.6.7 DVBHTMLFrameElement Interface

8.11.4.6.7.1 IDL Definition

```
interface DVBHTMLFrameElement : DVBHTMLElement {

    readonly attribute Document contentDocument;
    readonly attribute DOMString frameBorder; // Diverge from W3C DOM HTML
    attribute DOMString longDesc;
    readonly attribute DOMString marginHeight // Diverge from W3C DOM HTML
    readonly attribute DOMString marginWidth; // Diverge from W3C DOM HTML
    readonly  attribute DOMString scrolling; // Diverge from W3C DOM HTML
    attribute DOMString src;
};
```

The semantics of the attributes associated with this interface are the same as the semantics associated with the `HTMLFrameElement` interface defined in Document Object Model (DOM) Level 1 Specification [35], with the extensions, restrictions or amendments defined in clause 8.11.4.6.7.2.

8.11.4.6.7.2 Attributes

Table 44

Name	Readonly	Type	Description
frame Border	yes	DOMString	As defined in Document Object Model (DOM) Level 1 Specification [35]. This attribute has been moved to readonly.
margin Height	yes	DOMString	As defined in Document Object Model (DOM) Level 1 Specification [35]. This attribute has been moved to readonly.
margin Width	yes	DOMString	As defined in Document Object Model (DOM) Level 1 Specification [35]. This attribute has been moved to readonly.
scrolling	yes	DOMString	As defined in Document Object Model (DOM) Level 1 Specification [35]. This attribute has been moved to readonly.

8.11.4.6.8 DVBHTMLFrameSetElement Interface

```
8.11.4.6.8.1    IDL Definition
interface DVBHTMLFrameSetElement : DVBHTMLElement {
    readonly attribute DOMString cols; // Diverge from W3C DOM HTML
    readonly attribute DOMString rows; // Diverge from W3C DOM HTML
};
```

The semantics of the attributes associated with this interface are the same as the semantics associated with the `HTMLFrameSetElement` interface defined in Document Object Model (DOM) Level 1 Specification [35], with the extensions, restrictions or amendments defined in clause 8.11.4.6.8.2.

8.11.4.6.8.2 Attributes

Table 45

Name	Readonly	Type	Description
cols	yes	DOMString	As defined in Document Object Model (DOM) Level 1 Specification [35]. This attribute has been moved to readonly.
rows	yes	DOMString	As defined in Document Object Model (DOM) Level 1 Specification [35]. This attribute has been moved to readonly.

8.11.4.6.9 DVBHTMLIFrameElement Interface

8.11.4.6.9.1 IDL Definition

```
interface DVBHTMLIFrameElement : DVBHTMLElement {
attribute DOMString frameBorder;
attribute DOMString longDesc;
attribute DOMString scrolling;
attribute DOMString src;
attribute DVBHTMLDocument contentDocument;
};
```

The semantics of the attributes associated with this interface are the same as the semantics associated with the `HTMLIFrameElement` interface defined in Document Object Model (DOM) Level 1 Specification [35].

8.11.4.6.9.2 Attributes

Table 46: IFrame element attributes

Name	Readonly	Type	Description
content Document	no	DVBHTMLDocument	As defined in Document Object Model (DOM) Level 1 Specification [35]. Only the type differs.

8.11.4.6.10 DVBHTMLImageElement Interface

8.11.4.6.10.1 IDL Definition

```
interface DVBHTMLImageElement : DVBHTMLElement {
    attribute DOMString alt;
    attribute DOMString height; // Extension to W3C DOM HTML
    attribute DOMString longDesc;
    attribute DOMString lowSrc;
    attribute DOMString src;
    readonly attribute DOMString useMap; // Diverge from W3C DOM HTML
    attribute DOMString width;
};
```

The semantics of the attributes associated with this interface are the same as the semantics associated with the `HTMLImageElement` interface defined in Document Object Model (DOM) Level 1 Specification [35], with the extensions, restrictions or amendments defined in clause 8.11.4.6.10.2.

8.11.4.6.10.2 Attributes

Table 47

Name	Readonly	Type	Description
longdesc	no	DOMString	As defined in Document Object Model (DOM) Level 1 Specification [35], A DVB-HTML User Agent may optionally make use of this resource in device specific ways (e.g for accessibility). The externally referenced resource will not form part of the document model.
useMap	yes	DOMString	As defined in Document Object Model (DOM) Level 1 Specification [35]. This attribute has been moved to readonly. The attribute value shall be the id value and not the name value of the associated map element (since the name attribute in the map element is not supported in DVB-HTML).

8.11.4.6.11 DVBHTMLObjectElement Interface

8.11.4.6.11.1 IDL Definition

```
interface DVBHTMLObjectElement : DVBHTMLElement {
attribute DOMString archive;
attribute DOMString code;
attribute DOMString codeBase;
attribute DOMString codeType;
readonly attribute DVBHTMLDocument contentDocument; // Introduced in DVB-HTML:
attribute DOMString data;
attribute boolean declare;
readonly attribute DVBHTMLFormElement form;
attribute DOMString height;
attribute DOMString name;
attribute DOMString standby;
attribute DOMString type;
readonly attribute DOMString useMap;
attribute DOMString width;
};
```

The semantics of the attributes associated with this interface are the same as the semantics associated with the HTMLObjectElement interface defined in Document Object Model (DOM) Level 1 Specification [35], with the addition of the contentDocument attribute defined in clause 8.11.4.6.11.2.

8.11.4.6.11.2 Attributes

Table 48

Name	Readonly	Type	Description
contentDocument	yes	DOMString	The document this object contains, if there is any and it is available, or null otherwise.

8.11.4.6.12 DVBHTMLInputElement Interface

8.11.4.6.12.1 IDL Definition

```
interface DVBHTMLInputElement : DVBHTMLElement {

    attribute DOMString accept;
    attribute DOMString accessKey;
    attribute DOMString alt;
    attribute boolean checked; // Extension to W3C DOM HTML
    attribute DOMString defaultValue; // Extension to W3C DOM HTML
    attribute boolean defaultChecked; // Extension to W3C DOM HTML
    attribute boolean disabled; // Extension to W3C DOM HTML
    readonly attribute DVBHTMLFormElement form; // DVB-HTML Type coherence
    attribute long maxLength;
    attribute DOMString name;
    attribute boolean readOnly; // Extension to W3C DOM HTML
    attribute DOMString size;
    attribute DOMString src;
    readonly attribute DOMString type; // Diverge from W3C DOM HTML
    readonly attribute DOMString useMap; // Diverge from W3C DOM HTML
    attribute DOMString value; // Extension to W3C DOM HTML

    void blur();
    void click();
    void focus();
    void select();
};
```

The semantics of the attributes and methods associated with this interface are the same as the semantics associated with the `HTMLInputElement` interface defined in Document Object Model (DOM) Level 1 Specification [35], with the extensions, restrictions or amendments defined in clause 8.11.4.6.12.2.

8.11.4.6.12.2 Attributes

Table 49

Name	Readonly	Type	Description
checked	no	boolean	As defined in Document Object Model (DOM) Level 1 Specification [35]; this attribute corresponds to the current value of controls (as defined in clause 8.11.4.3.2 "Initial and current values of form controls") with attribute type equals to checkbox or radio. It shall be ignored for the other types of control.
default Checked	no	boolean	As defined in Document Object Model (DOM) Level 1 Specification [35]; this attribute corresponds to the initial value of controls (as defined in clause 8.11.4.3.2 "Initial and current values of form controls") with attribute type equals to checkbox or radio. It is initialized by the checked attribute of the <input> tag. It shall be ignored for the other types of control.
default Value	no	DOMString	As defined in Document Object Model (DOM) Level 1 Specification [35]; this attribute corresponds to the initial value of controls (as defined in clause 8.11.4.3.2 "Initial and current values of form controls") with attribute type equals to text or password. It is initialized by the value attribute of the <input> element.
disabled	no	boolean	As defined in Document Object Model (DOM) Level 1 Specification [35]. If set to true, the effects are the following: • Input does not receive the focus. • Input is not included in the tabbing navigation. • Input is not successful. Since the readOnly state is less restrictive than the disabled state, if the disabled attribute is set to true, the control evolves to the disabled state, independently of the readOnly attribute value. If the disabled attribute is set to false, the control will evolve to the readOnly state depending on the value of the readOnly attribute. By default, the attribute value is FALSE.
form	yes	DVBHTMLFormElement	As defined in Document Object Model (DOM) Level 1 Specification [35], only the return type differs.
readOnly	no	boolean	As defined in Document Object Model (DOM) Level 1 Specification [35]. When set to true and textarea is not in the disabled state, the effects are the following : Input receives the focus but cannot be modified by the user. Input is included in the tabbing navigation. Input is successful. Since the readOnly state is less restrictive than the disabled state, if the readOnly attribute is set to true when the disabled attribute is equal to true, the control remains in the disabled state. By default, the attribute value is `FALSE`.
type	yes	DOMString	As defined in Document Object Model (DOM) Level 1 Specification [35]. This attribute has been moved to readonly.
useMap	yes	DOMString	As defined in Document Object Model (DOM) Level 1 Specification [35]. This attribute has been moved to readonly. The attribute value shall be the id value and not the name value of the associated map element (since the name attribute in the map element is not supported in DVB-HTML).
value	no	DOMString	As defined in Document Object Model (DOM) Level 1 Specification [35]. This attribute corresponds to the current value of controls (as defined in clause 8.11.4.3.2, "Initial and current values of form controls" with attribute type equals to text, password, hidden, submit, reset, button, image, or file. For all the controls, it represents the value of the name/value pair sent to the server when the form containing this control is submitted. This attribute is initialized by the value attribute of the <input> tag.

8.11.4.6.13 DVBHTMLOptionElement Interface

This interface models the `<option>` element.

8.11.4.6.13.1 IDL Definition

```
interface DVBHTMLOptionElement : DVBHTMLElement {
    attribute boolean defaultSelected; // Extension to W3C DOM HTML
    attribute boolean disabled;
    readonly attribute DVBHTMLFormElement form; // DVB-HTML Type coherence
    readonly attribute long index;
    attribute DOMString label;
    attribute boolean selected; // Extension to W3C DOM HTML
    readonly attribute DOMString text;
    attribute DOMString value;
};
```

The semantics of the attributes associated with this interface are the same as the semantics associated with the `HTMLOptionElement` interface defined in Document Object Model (DOM) Level 1 Specification [35], with the extensions, restrictions or amendments defined in clause 8.11.4.6.13.2.

8.11.4.6.13.2 Attributes

Table 50

Name	Readonly	Type	Description
default Selected	no	boolean	As defined in Document Object Model (DOM) Level 1 Specification [35]. The initial value is as defined in clause 8.11.4.3.2, "Initial and current values of form controls". If the option is included in a single choice select, it is initialized to true if the attribute checked is present in the option tag and if this option is the latest option between those contained in the select with the attribute checked set. Its value is false otherwise. If the option is included in a multiple choice select, it is initialized to true if the attribute checked is present in the option tag, false otherwise.
form	yes	DVBHTMLFormElement	As defined in Document Object Model (DOM) Level 1 Specification [35]. Only the return type differs
selected	yes	boolean	As defined in Document Object Model (DOM) Level 1 Specification [35]. The current value is as defined in clause 8.11.4.3.2, "Initial and current values of form controls".

8.11.4.6.14 DVBHTMLSelectElement Interface

This interface represents the `<select>` control which allows the selection of one or multiple options among a list. Some graphical user agents represent single and multiple selection with different types of widgets. Options can be grouped inside an option group, for presentational reasons, or directly inserted in the selection. Option groups can not be nested.

8.11.4.6.14.1 IDL Definition

```
interface DVBHTMLSelectElement : DVBHTMLElement {
    attribute boolean disabled;
    readonly attribute DVBHTMLFormElement form; // DVB-HTML Type coherence
    readonly attribute unsigned long length; // Lengths type coherence
    readonly attribute boolean multiple; // Diverge from W3C DOM HTML
    attribute DOMString name;
    readonly attribute DVBHTMLCollection options; // DVB-HTML Type coherence
    attribute long selectedIndex;
                    // Extension to W3C DOM HTML
    attribute unsigned long size; // Size type coherence
    readonly attribute DOMString type;
    attribute DOMString value;
```

```
    void add(in DVBHTMLElement element, in DVBHTMLElement before) raises(DOMException);
    void blur();
    void focus();
    void remove(in long index);
};
```

The semantics of the attributes and methods associated with this interface are the same as the semantics associated with the `HTMLSelectElement` interface defined in Document Object Model (DOM) Level 1 Specification [35], with the extensions, restrictions or amendments defined in clause 8.11.4.6.14.2.

8.11.4.6.14.2 Attributes

Table 51

Name	Readonly	Type	Description
form	yes	DVBHTMLFormElement	As defined in Document Object Model (DOM) Level 1 Specification [35]. Only the return type differs.
length	yes	unsigned long	As defined in Document Object Model (DOM) Level 1 Specification [35]. Only the return type differs.
multiple	yes	boolean	As defined in Document Object Model (DOM) Level 1 Specification [35]. This attribute has been moved to readonly.
options	yes	DVBHTMLCollection	As defined in Document Object Model (DOM) Level 1 Specification [35]. Only the return type differs.
selected Index	no	long	The ordinal index of the selected option, starting from 0. The value -1 is returned if no element is selected. If multiple options are selected, the index of the first selected option is returned. Upon modification, the display shall be affected.
size	no	unsigned long	As defined in Document Object Model (DOM) Level 1 Specification [35]. Only the return type differs.

When a document is trying to access the value of the `options` attribute before the parsing of the contents of the `select` element is complete, the returned value shall be the collection of matching elements at the time of the call.

8.11.4.6.15 DVBHTMLTextAreaElement Interface

This interface represents a multiline text area. It refers to the `<textarea>` DVB-HTML tag.

8.11.4.6.15.1 IDL Definition

```
interface DVBHTMLTextArea : DVBHTMLElement {
    attribute DOMString accessKey;
    readonly attribute unsigned long cols; // Diverge from W3C DOM HTML
    attribute DOMString defaultValue;
                    // Extension to W3C DOM HTML
    attribute boolean disabled; // Extension to W3C DOM HTML
    readonly attribute DVBHTMLFormElement form; // DVB-HTML Type coherence
    attribute DOMString name;
    attribute boolean readOnly // Extension to W3C DOM HTML
    readonly attribute unsigned long rows; // Diverge from W3C DOM HTML
    readonly attribute DOMString type;
    attribute DOMString value; // Extension to W3C DOM HTML

    void blur();
    void focus();
    void select();
}
```

The semantics of the attributes and methods associated with this interface are the same as the semantics associated with the `HTMLTextAreaElement` interface defined in Document Object Model (DOM) Level 1 Specification [35], with the extensions, restrictions or amendments defined in clause 8.11.4.6.15.2.

8.11.4.6.15.2 Attributes

Table 52

Name	Readonly	Type	Description
cols	yes	unsigned long	As defined in Document Object Model (DOM) Level 1 Specification [35]. Only the return type differs and it has been moved to readonly.
default Value	no	DOMString	As defined in Document Object Model (DOM) Level 1 Specification [35]. The initial value (as defined in clause 8.11.4.3.2, "Initial and current values of form controls") is initialized by the text inserted between the start and end tags of the element.
disabled	no	boolean	As defined in Document Object Model (DOM) Level 1 Specification [35]. If set to true, the effects are the following : • Textarea does not receive the focus. • Textarea is not included in the tabbing navigation. • Textarea is not successful. Since the readOnly state is less restrictive than the disabled state, if the disabled attribute is set to true, the control evolves to the disabled state, independently of the readOnly attribute value. If the disabled attribute is set to false, the control will evolve to the readOnly state depending on the value of the readOnly attribute. By default, the attribute value is FALSE.
form	yes	DVBHTMLFormElement	As defined in Document Object Model (DOM) Level 1 Specification [35]. Only the return type differs.
read Only	no	boolean	As defined in Document Object Model (DOM) Level 1 Specification [35]. If set to true and textarea is not in the disabled state, the effects are the following : • Textarea receives the focus but cannot be modified by the user. • Textarea is included in the tabbing navigation. • Textarea is successful. Since the readOnly state is less restrictive than the disabled state, if the readOnly attribute is set to true when the disabled attribute is equal to true, the control remains in the disabled state. By default, the attribute value is `FALSE`.
rows	yes	unsigned long	As defined in Document Object Model (DOM) Level 1 Specification [35]. Only the return type differs and it has been moved to readonly.
value	no	DOMString	As defined in Document Object Model (DOM) Level 1 Specification [35]. The current value is as defined in clause 8.11.4.3.2, "Initial and current values of form controls".

8.11.5 DVB Exceptions

DOM operations may raise exceptions in "exceptional" circumstances. In general, DVB-HTML will raise exceptions defined by Document Object Model (DOM) Level 2 Core Specification [34], however certain methods in DVB-HTML raise other exceptions under other circumstances.

In DVB-HTML the `DVBException` is defined, this is in addition to the DOMException of DOM 2 and defines additional error codes.

NOTE: The error codes defined for DVBException are defined so as not to clash with those defined for DOMException, however the W3C retains the right to define additional codes for DOMException which may conflict with DVBException in the future. To avoid conflicts the programmer may use typeof: the DVBException, as a host object, has a defined type different to that of DOMException.

8.11.5.1 DVBException

The value returned by the **typeof** operator for DVBException shall be "**dvbexception**".

8.11.5.1.1 IDL Definition

```
exception DVBException {
unsigned short code;
};

// ExceptionCode
// Introduced in DVB-HTML:
const unsigned short NON_CONFORMANT_ERR = 601;
const unsigned short SECURITY_VIOLATION_ERR = 602;
const unsigned short WINDOW_NOT_FOUND_ERR = 603;
const unsigned short SUBCLASS_NOT_ALLOWED_ERR = 604;
const unsigned short CONVERSION_TYPE_ERR = 605;
```

ExceptionCode: An integer indicating the type of error generated.

Other numeric codes are reserved by DVB for possible future use.

8.11.5.1.2 Defined Constants

NON_CONFORMANT_ERR: An operation attempted to alter the DOM in such a way as would make the document non-conformant.

SECURITY_VIOLATION_ERR: An operation failed because the application did not have sufficient permission to perform it.

WINDOW_NOT_FOUND_ERR: The application attempted to open a window which was not part of the application.

SUBCLASS_NOT_ALLOWED_ERR: The application attempted to subclass a final class.

CONVERSION_TYPE_ERR: The application attempted to subclass a final class.

8.11.6 Language bindings

8.11.6.1 ECMAScript Binding

See annex AC, "ECMAScript Binding".

8.11.6.2 Java Binding

See annex AD, "Support for DVB-HTML".

8.11.7 DVB Environment object module

A DOM application can use the hasFeature method of the DOM implementation interface to determine that this module is supported. The feature string for all interfaces listed in this module is "DVBEnvironment" and the version "2.0".

8.11.7.1 Free variables

DVB-HTML shall support ECMAScript access to certain APIs is through a number of variables, which get bound in each page context:

- document;
- navigator;
- window.

Each DVB-HTML application shall see different Navigator, Document and Window objects, Only one Navigator object is created per application lifetime (therefore properties may be set on the Navigator object which persist for the lifetime of the application). The navigator variable is bound to this object in each page context.

Windows and Document are initialized on an as needed basis, Document is valid during and after parsing of the document and is bound to the variable document.

Each window is associated with a single document, there may be more than one extant window object per DVB-HTML application, however multiple window support is included only for frames. There shall be at most one top level window per DVB-HTML application (see CSS 2 [39]).

The window variable is initialized to the immediately containing window for a document.

8.11.7.2 Environmental host objects

8.11.7.2.1 Navigator Object

Represents the browser itself, the navigator free variable is of this type. This is supported in DVB-HTML with the following attributes and methods.

8.11.7.2.1.1 IDL Definition

```
interface Navigator {
    readonly attribute String appCodeName;
    readonly attribute String appName;
    readonly attribute String appVersion;
    readonly attribute String userAgent;
};
```

8.11.7.2.1.2 Attributes

Table 53: Navigator Object Attributes

appCodeName	Readonly	Specifies the code name of the browser with the value specified in the BNF in clause 8.13.1, "User agent strings".
appName	Readonly	Specifies the name of the browser, the value shall be "DVB-HTML".
appVersion	Readonly	Specifies version information for the Navigator with the value specified in the BNF in clause 8.13.1, "User agent strings".
userAgent	Readonly	Specifies the user agent header see clause 8.13.1, "User agent strings".

8.11.7.2.2 Window object

Represents a top-level browser window or a frame. There is only one top-level window per DVB-HTML application, (i.e. whose parent is NULL). Other window objects in an application represent frames in a frameset.

The top level Window object of an application is created in the CSS initial containing block for the application. The viewport and containing block of the application are controlled by the stylesheet associated with the current document held by the top level Window object. Other Window objects are shaped by the frameset rules.

Window is supported in DVB-HTML with the following attributes and methods.

8.11.7.2.2.1 IDL Definition

```
interface Window {
    readonly attribute DVBHTMLDocument document;
    readonly attribute DOMImplementation implementation;
    readonly attribute DVBHTMLCollection frames;
    readonly attribute unsigned long length;
    attribute Location location;
    readonly attribute String name;
    readonly attribute Navigator navigator;
    readonly attribute Window parent;
    readonly attribute Window window;
    readonly attribute Window self;
    attribute String status;
    attribute String defaultStatus;
    readonly attribute Window top;
    void clearTimeout(in unsigned long timerId);
    Window open(in String url, in String name, in String features);
    unsigned long setTimeout(in String statement, in unsigned long delay);
}
```

8.11.7.2.2.2 Attributes

The following attributes are supported.

Table 54: Window Object Attributes

Attribute	Modifier	Semantics
closed	Readonly	Specifies whether a window has been closed.
document	Readonly	Contains information on the current document. Returns an object of type DVBHTMLDocument.
implementation	Readonly	Supplies the method to bootstrap into the DOM interfaces.
frames	Readonly	An array reflecting all the frames in a window containing documents from the current application. Frames that belong to another application shall not appear in the collection.
location	Readwrite	Contains information on the URL used to request the current document. Setting the value of window.location sets the href value of the window's associated location object, and causes the window to navigate to the URL. NOTE 1: The **requested** URL is stored in window.location and this may be different than the **actual** URL of the delivered document, which is stored in document.URL. If a string is used to set the location field, then it is converted to a Location. If the string is not a well formed URL, or the user agent is unable to navigate to it the location is unchanged. Setting this string to the "exit:" locator (see clause 8.16.5.2, "Exit locator") shall cause termination as defined in clause 9.3.3.5, "Killed".
name	Readwrite	A unique name used to refer to this window.
length	Readonly	the length of the frames collection
parent	Readonly	Returns the Window whose frameset contains the current frame or NULL if the window is the outer most window. A frame may return NULL for this attribute for security reasons (see clause 8.14.3, "Inter application security").
status	Readwrite	Specifies a priority or transient message in the window's status bar. NOTE 2: It is not required this actually be displayed anywhere - but calling it must cause no harm).
defaultStatus	Readwrite	The default message for the status bar. NOTE 3: It is not required this actually be displayed anywhere - but calling it must cause no harm.
top	Readonly	A synonym for the topmost browser window. The value of this attribute shall be NULL if the referenced frame is not available for security.
self	Readonly	Returns the current Window
document	Readonly	Returns the document instance the window is displaying. If the document is inaccessible due to security reasons (see clause 8.14.3 "Inter application security") then shall return NULL.
navigator	Readonly	Returns the Navigator object for the application.

8.11.7.2.2.3 Methods

The following methods are supported.

Table 55: Window open method

Method	Name	Description		
	open	Sets the document of the specified window to a new resource.		
Parameters	**Name**	**Type**	**Qualifier**	**Description**
	url	String	in	The location of the document to display in the new window.
	name	String	in	The target frame within the frameset or the top level window.
	features	String	in	unused
Return value	**Type**	**Description**		
	Window	This method returns the Window object in which the document has been opened.		

Table 56: Window setTimeout method

Method	Name	Description		
	setTimeout	Evaluates an expression or calls a function once after a specified number of milliseconds has elapsed. When the document is replaced then all timeouts on the window are cleared.		
Parameters	**Name**	**Type**	**Qualifier**	**Description**
	statement	String	in	The piece of ECMAScript to be evaluated.
	delay	unsigned long	in	The length of time to wait before executing the statement in milliseconds.
Return value	**Type**	**Description**		
	unsigned long	This method returns a timerId that uniquely identifies the timer.		

Table 57: Window clearTimeout method

Method	Name	Description		
	clearTimeout	Cancels a timeout that was set with the setTimeout method. When the document is replaced then all timeouts on the window are cleared.		
Parameters	**Name**	**Type**	**Qualifier**	**Description**
	timerId	unsigned long	in	The id of the timer to clear as returned by setTimount.
Return value	**Type**	**Description**		
	none	This method does not return a value.		

8.11.7.2.3 Location object.

A convenience element that parses URIs. Is used in the location field of the window object.

8.11.7.2.3.1 IDL Definition

```
interface Location {
attribute DOMString hash;
attribute DOMString host;
attribute DOMString hostname;
attribute DOMString href;
attribute DOMString pathname;
attribute DOMString port;
attribute DOMString protocol;
attribute DOMString search;
};
```

8.11.7.2.3.2 Attributes

Table 58: Location Object Attributes

Attribute	Modifier	Semantics
hash	Readwrite	The portion of the URI after and including the '#' symbol.
host	Readwrite	The hostname and port of the URI.
hostname	Readwrite	The hostname (without the port) of the URI.
href	Readwrite	The complete URI.
pathname	Readwrite	The pathname portion of the URI.
port	Readwrite	The port of the URI.
protocol	Readwrite	The protocol portion of the URI.
search	Readwrite	The portion of the URI after and including the '?' symbol, not including the hash portion.

8.11.8 CSS Support

8.11.8.1 DVB CSS DOM module

DVB-HTML does not require support for the DOM-2 CSS modules, however it does require support of the this simpler CSS module. A DOM application can use the hasFeature method of the DOM implementation interface to determine that this module is supported. The feature string for all interfaces listed in this module is "DVBCSS" and the version "2.0".

8.11.8.1.1 DVBCSSInlineStyle

DVBInlineStyle: This interface shall be implemented by objects that implement the Element interface (see Document Object Model (DOM) Level 2 Core Specification [34]), and which represent elements that support the style attribute.

A DVB-HTML user agent may support the DOM-2 CSS module, in which case this interface shall be replaced by ElementCSSInlineStyle, however in such a case the CSS2Properties interface shall also be implemented, and the object returned by the style attribute on ElementCSSInlineStyle shall also implement the CSS2Properties interface.

8.11.8.1.1.1 IDL Definition

```
// Introduced in MHP1.1:
interface DVBCSSInlineStyle {
readonly attribute CSS2Properties style;
};
```

8.11.8.1.1.2 Attributes

style: of type CSS2Properties as defined in section 2.3 of Document Object Model (DOM) Level 2 Style Specification [37].

8.11.8.1.2 DVBCSSStyle

DVBCSSStyle: This interface shall be implemented by objects that implement the Element interface (see Document Object Model (DOM) Level 2 Core Specification [34]), that represent the root element of a document.

8.11.8.1.2.1 IDL Definition

```
// Introduced in MHP1.1:
interface DVBCSSStyle {
readonly attribute DVBCSSViewportRule viewport;
};
```

8.11.8.1.2.2 Attributes

viewport: of type DVBCSSViewportRule. If the document containing the element is not in a root window this attribute shall be NULL, if it is in the root window then this attribute shall return an object that implements the DVBCSSViewportRule interface.

8.11.8.1.3 DVBCSSViewportRule

DVBCSSViewportRule: The DVBCSSviewportRule interface represents a @viewport rule within a CSS style sheet. The @viewport rule is used to specify the on screen regions and relationship to video of an DVB-HTML application.

8.11.8.1.3.1 IDL Definition

```
// Introduced in MHP 1.1:
interface DVBCSSViewportRule : CSSRule {
    readonly attribute DVBCSSViewportProperties style;
};
```

8.11.8.1.3.2 Attributes

style: of type DVBCSSViewportProperties, readonly. An object that represents all the properties of the active viewport.

8.11.8.1.4 DVBCSSViewportProperties

DVBCSSViewportProperties: The DVBCSSViewportProperties interface represents a mechanism for retrieving and setting properties within a viewport. The attributes of this interface correspond to all the properties specified in clause 8.8.5.3.2, "The @dvb-viewport rule".

8.11.8.1.4.1 IDL Definition

```
// Introduced in MHP1.1:
interface DVBCSSViewportRule {
readonly attribute DOMSting pseudoClass;
attribute DOMString area;
attribute DOMString[] backgroundVideoTransform;
attribute DOMString[] backgroundVideo;
attribute DOMString[] backgroundVideoPreserveAspect;
attribute DOMString[] backgroundVideoClip;
attribute DOMString[] backgroundVideoRectangle;
attribute DOMString background;
attribute DOMString backgroundImageRectangle;
attribute DOMString initial;
attribute Number verticalResolution;
attribute Number horizontalResolution;
attribute DOMString scene;
};
```

8.11.8.1.4.2 Attributes

Table 59: Location Object Attributes

Attribute	Modifier	Semantics
pseudoClass	Readonly	See the pseudoClass property definition in clause 8.8.5.3.2, "The @dvb-viewport rule". Exception: NO_MODIFICATION_ALLOWED_ERR: Raised on any attempt to set this property.
area	Readwrite	See the area property definition in clause 8.8.5.3.2, "The @dvb-viewport rule". Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparsable.
backgroundVideo Transform	Readwrite	See the backgroundVideoTransform property definition in clause 8.8.5.3.2, "The @dvb-viewport rule". Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparsable (see note).
backgroundVideo	Readwrite	See the backgroundVideo property definition in clause 8.8.5.3.2, "The @dvb-viewport rule". Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparsable (see note).
backgroundVideo PreserveAspect	Readwrite	See the backgroundVideoPreserveAspect property definition in clause 8.8.5.3.2, "The @dvb-viewport rule". Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparsable (see note).
backgroundVideoClip	Readwrite	See the backgroundVideoClip property definition in clause 8.8.5.3.2, "The @dvb-viewport rule". Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparsable (see note).
backgroundVideo Rectangle	Readwrite	See the backgroundVideoRectangle property definition in clause 8.8.5.3.2, "The @dvb-viewport rule". Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparsable (see note).
background	Readwrite	See the background property definition in clause 8.8.5.3.2, "The @dvb-viewport rule". Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparsable (see note).
backgroundImage Rectangle	Readwrite	See the backgroundImageRectangle property definition in clause 8.8.5.3.2, "The @dvb-viewport rule". Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparsable (see note).
initial	Readwrite	See the initial property definition in clause 8.8.5.3.2, "The @dvb-viewport rule". Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparsable.
verticalResolution	Readwrite	See the verticalResolution property definition in clause 8.8.5.3.2, "The @dvb-viewport rule". Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparsable.
horizontalResolution	Readwrite	See the horizontalResolution property definition in clause 8.8.5.3.2, "The @dvb-viewport rule". Exception: SYNTAX_ERR: Raised if the new value has a syntax error and is unparsable.
NOTE: This attribute is an array of values, one for each defined background plane.		

8.12 Cookie support

8.12.1 DOM Cookie Interface

The cookie attribute of the document (see clause 8.11.4.5.2 "DVBHTMLDocument Interface") provides an interface to the set of attribute-value pairs described in RFC 2109 [43].

The cookie attribute of document is a read/write string with the following behaviour:

a) The attribute may be assigned a string value `set-cookie` obeying the following BNF, where ALPHA and DIGIT are as specified in RFC 2616 [17]:

```
set-cookie   = cookie
cookie       = av-pair
         ["; expires=" [ rfc1123-date ] ]
         ["; path=" [ path ] ]
         ["; domain=" [ domain] ]
         ["; secure" ]
         ["; dvb-scope=" [ dvb-scope ]]
av-pair      = attr ["=" value ]
attr         = 1*(ALPHA | DIGIT | aspecials)
aspecials    = ( "!" | "#" | "$" | "&" | "'" | "*" | "+" | "-"
```

```
                | "." | "_" | "`" | "|" | "~" | "%" )

value       = 1*(ALPHA | DIGIT | special | vspecials)
vspecials   = "(" | ")" | "<" | ">" | "@"           | "," | ":" | "\" | <"> | "/"
                | "[" | "]" | "?" | "="             | "{" | "}" | SP  | HT

path        = 1*(ALPHA | DIGIT | aspecials)
domain      = 1*(ALPHA | DIGIT | aspecials)
dvb-scope   = "App" | "Org" | "All"

 rfc1123-date is the subset of the RFC 1123 [44] date format as specified in RFC 2616 [17]
(HTTP 1.1).
rfc1123-date        = wkday ", " SP date1 SP time " GMT"
wkday               = "Mon" | "Tue" | "Wed" | "Thu" | "Fri" | "Sat" | "Sun"
date1               = 2DIGIT SP month SP 4DIGIT
time                =   2DIGIT ":" 2DIGIT ":" 2DIGIT
month               =    "Jan" | "Feb" | "Mar" | "Apr"| "May" | "Jun"
                         | "Jul" | "Aug" | "Sep" | "Oct"| "Nov" | "Dec"
```

b) If an expiration is not specified, the lifetime of the cookie is that of the DVB-HTML application.

c) When read the attribute returns a string `get-cookie` consisting of the av-pairs for all cookies that have not expired and are within the scope of the current document.

```
get-cookie          =   av-pair *(";" av-pair)
```

NOTE 1: If the same attribute is specified for multiple paths that apply to the current document, the attribute appears in multiple av-pairs.

NOTE 2: "%" is a legal character in values. As defined in RFC 2109 [43] characters not allowed in values may be escaped using "% HEX HEX", there is no special handling of this escaping by the ECMAScript API.

8.12.2 Cookie Storage and Lifetime

8.12.2.1 Cookie Storage Limits

An implementation should attempt to meet the minimum implementation limits of RFC 2109 [43] including (although not necessarily simultaneously) at least 20 cookies of 4 096 bytes. The size is the total required for the cookies and all attributes stored as strings.

8.12.2.2 Cookie Persistence

If the application requests permission for access to persistent storage, this is granted and persistent storage is available, an implementation should attempt to store cookies until they expire. If the application does not have permission for persistent storage or the storage is not available then cookies shall persist only for the lifetime of the DVB-HTML application.

8.12.2.3 Privacy Considerations

Notwithstanding any other requirements in this clause, an implementation may provide mechanisms for controlling the extent of cookie support, including which, if any, cookies are stored, temporarily or persistently.

8.12.3 Cookie Scoping

8.12.3.1 General Rules

All cookies are scoped by dvb-scope. The dvb-scope limits cookies to being accessible by pages in applications of a particular organization (dvb-scope=Org), by pages with a particular Application Identifier (dvb-scope=App) or by pages within all applications (dvb-scope=All). When not specified, the dvb-scope defaults to App. Cookies of unsigned applications are always scoped to App. Cookies with different scoping are stored separately and reported separately both through the DOM and when transmitted to a server. Setting a cookie with the same scoping as an existing one causes the replacement of the existing one.

8.12.3.2 Documents delivered via DSM-CC Object Carousel

Cookies set via the DOM on pages delivered by DSM-CC carousel transport are never sent back to a server, even when http: URLs are within the application boundary. The domain, path and secure attributes are ignored for such cookies.

8.12.3.3 Documents delivered via HTTP transport

Cookies associated with pages delivered by http: shall be scoped by domain and path in addition to scoping by their dvb-scope. If the domain is specified, the cookie is only accessible to pages delivered from hosts within that domain. The default domain is the host name of the http server which delivered the page.

If the path is specified, the associated cookie is only accessible to pages in directories under that path.

Domain and path matching are performed per RFC 2109 [43] with the exception that a host match is generated if either the domain string or the domain string with a period prepended to it would generate a match with the basic algorithm.

NOTE: the modified matching algorithm provides both domain compatibility with cookies whose domain is set from the DOM or from HTTP 1.0 or 1.1 servers that only support the protocol in clause 8.12.1, "DOM Cookie Interface". In both cases, leading dots are frequently left off.

Setting a cookie with a domain or path for which the current page is out of scope has no effect and the cookie is discarded.

Cookies with the secure attribute specified shall only be sent over secure, HTTPS connections.

8.12.4 HTTP Cookie Support

8.12.4.1 Background

The MHP profile of HTTP 1.0 profile allows both HTTP 1.0 and HTTP 1.1 transport (see clause 6.3.7.2, "MHP profile of HTTP 1.0"). In order for cookie transport to work with existing HTTP 1.0 servers, support for "Netscape" cookie receipt is required. Cookie transmission can always be done with RFC 2109 [43] format since most servers will consider the $Version and $Path attributes as unknown cookies and ignore them.

8.12.4.2 Sending Cookies

The client shall send cookies per RFC 2109 [43] and for compatibility with older servers, DVB-HTML implementations should use semi-colons to separate cookies rather than commas (both are allowed by RFC 2109 [43]). Only cookies in the scope of the document, including any dvb-scope restrictions, are sent.

The client shall accept cookies via a Set-Cookie header in the HTTP response. The format of the contents of that header may be either in RFC 2109 [43] format or in that specified by the set-cookie BNF from the DOM description above (see clause 8.12.1, "DOM Cookie Interface").

8.12.4.3 Receiving Cookies

The client shall accept cookies via a Set-Cookie header in the HTTP response. The format of the contents of that header may be either in RFC 2109 [43] format or in "Netscape" cookie format, where Netscape format is that specified by the set-cookie BNF from the DOM description above.

8.13 HTTP User Agent String Support

Requests from the DVB-HTML user agent shall include an HTTP header of the form:

```
User-Agent: <user-agent>
```

where <user-agent> is the DVB-HTML user agent string.

8.13.1 User agent strings

8.13.1.1 Current user agent-related strings

Current examples (from the internet) of the user agent string are shown in table 60.

Table 60: Internet user agent string examples

Field	Netscape Navigator	Internet Explorer on Win2K
userAgent	Mozilla/4.7 [en] (WinNT; U)	Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
appCodeName	Mozilla	Mozilla
appVersion	4.7 [en] (WinNT; U)	4.0 (compatible; MSIE 5.01; Windows NT)
appName	Netscape	Internet Explorer

The last three of these appear as properties on the navigator object. The properties `appCodeName` and `appVersion` are derived from the user agent string as reflected in the BNF. The user agent string generally has the BNF format. The `appName` is independent of the user agent string and its value shall always be "DVB-HTML" (see clause 8.11.7.2.1, "Navigator Object").

8.13.1.2 User agent string BNF

Adopting "Mozilla" as the appCodeName provides no interoperability benefit, and in fact would be incorrect as the DVB HTML user agent is not required to be Mozilla compliant. In general, the user agent string is best kept short, since it is sent with every HTTP request, hence it is reasonable to use abbreviations where appropriate.

The following BNF illustrates the user agent string that should be used:

```
userAgent ::= appCodeName "/" appVersion
appCodeName::= "DVB-HTML"
appVersion ::= appVersionNumber " (" identifiers ")"
appVersionNumber::= major "." minor "." micro
major::= [0-9]+
minor::= [0-9]+
micro::= [0-9]+
identifiers::= [ profileIdentifier ]+ [ "; " mhpIdentifier ]*
profileIdentifier ::= profileName " " major "." minor "." micro
profileName::= "eb" | "ib" | "ia"
mhpIdentifier::= [^;()]+
```

The syntax for the `major`, `minor` and `micro` elements are equal to the properties with the same names defined in GEM [1], clause 11.9.3, "Profile and version properties". The values for these properties are specified in clause 16.1, "System constants". mhpIdentifiers may be used to indicate optional features in the MHP implementation.

For example:

```
DVB-HTML/1.1.0 (eb 1.1.0; ipm)
```

Indicates an enhanced broadcast profile, with optional ip multicast support.

8.14 Security of DVB-HTML applications

8.14.1 Authentication of DVB-HTML files

DVB-HTML is subject to the same security model as DVB-J applications, that is a permissions file is associated with authenticated applications; unauthenticated applications operate within a sandbox environment, which means certain forms of locators will not operate, and certain APIs will not be available to ECMAScript.

Authenticated applications may be granted permission to use the restricted locators and APIs.

User Agents shall interpreting signed applications shall only make available the following file types to that application if they are signed by the same leaf certificate:

- DVB-HTML.
- ECMAScript.
- CSS.
- Inner application files (for DVB-J class files see GEM [1], clause 11.2.3, "Class Loading").

Locators that are disabled due to security restrictions will behave as if the resource pointed to is unavailable.

ECMAScript API calls that fail due to security restrictions will generate the DVBException with the error code SECURITY_VIOLATION_ERR.

8.14.2 Runtime code extension

8.14.2.1 Security principles

In order to address the security issues that arise with runtime code extension:

a) ECMAScript platforms shall permit the use of any of the ECMAScript runtime code extension techniques listed below for unauthenticated applications or authenticated applications running in the sandbox.

b) ECMAScript platforms shall permit the restricted use of runtime code extension techniques for authenticated applications running outside of the sandbox, where by restricted use is meant that the input to runtime code extension derives only from an appropriately authenticated source (see "Extensions to ECMAScript for trusted executable code").

c) ECMAScript platforms shall not permit the removal of pre-defined properties of host objects.

d) ECMAScript platforms shall not permit the replacement of a pre-defined function property of a host object.

8.14.2.1.1 Uses of runtime code extension in ECMAScript

The ECMAScript language supports runtime code extension by means of the following separate mechanisms:

- The `eval()` function property of the global object.
- The `Function` object.

In addition, the following mechanisms support runtime code extension through the following host object (platform) facilities:

- The `write()` function property of the DVB-HTML Document host object.
- The setTimeout(in String statement, in unsigned long delay) function property of the Window object.

The rules covering runtime code extension are detailed in clause 8.14.2.4, "Use of strings in RCEs".

8.14.2.2 Extensions to ECMAScript for trusted executable code

This clause details DVB specific mechanisms to allow Runtime Code Extension (RCE) features specified by the ECMAScript language specification and the allowed DOM APIs to be used without compromising security. In DVB-HTML additional mechanisms that keep track of the origin of strings in the system and disallow the potentially dangerous use in RCEs by privileged applications of strings from unverified external sources.

8.14.2.2.1 Propagation of Internal (safe) vs. External (unsafe) strings

This approach to providing security for the use of ECMAScript maintains application compatibility by distinguishing between value of an internal string type, a string value coming from within the ECMAScript language or literal strings within the application, and an external string type, a string value coming from (suspect) host-based objects. Both string values appear to content as ordinary ECMAScript string values to improve compatibility with the behaviour of DVB-HTML content in standard Internet browsers. This is accomplished by modifying the ECMAScript language specification to split the string type into two internal subtypes: *internal-string* and *external-string*. The difference between these subtypes is not visible to the application programmer, except in so far as a security exception may be thrown as a result of the use of some strings in an RCE.

This approach has the advantage of maintaining content compatibility with ECMAScript content written for existing browsers and to the W3C and ECMA specifications. In particular:

a) It allows DVB-HTML to conform to the W3C specified DOM APIs by allowing them to return values of type String rather than of type String Object.

b) It minimizes any increase in the runtime data footprint of ECMAScript applications by allowing the continued use of the much more compact String type, rather than String Object.

c) It minimizes any implementation change for DOM bindings between a W3C-complaint version and a DVB-HTML compliant version, since both can return String types.

d) It eliminates compatibility problems that would otherwise require changing core ECMAScript behaviour if String Objects were used (e.g. that two different instances of an object never test true for equality).

e) It allows finer-grained propagation of unsafe typing, since every string entity in the ECMAScript runtime would be tagged as safe (internal) or unsafe (external).

8.14.2.2.2 Modifying ECMA-262 to support Internal and External strings

This clause normatively specifies modifications to ECMA-262 [33] to differentiate the two kinds of string values.

The present document uses the editing convention that additions to ECMA-262 [33] are given in *italics*. Text that is removed is presented as ~~strike-out~~. When a change is made, the old text is given as ~~strike-out~~ and the new version in *italics*. The bold and bold-italics font faces are preserved from the ECMAScript standard document.

Change the last sentence of the second paragraph of clause 4.2 to read:

> A primitive value is a member of one of the following built-in types: **Undefined, Null, Boolean, Number,** ~~and String~~ *Internal-String, and External-String*; an object is a member of the remaining built-in type **Object**; and a method is a function associated with an object via a property. *The term String type is used when either of the types Internal-String or External-String is acceptable. The term String value is used when a value of either of the types Internal-String or External-String is acceptable.*

Change the first sentence of the first paragraph of clause 4.3.2 to read:

> A ***primitive value*** is a member of one of the types **Undefined**, **Null**, **Boolean**, **Number**, ~~and **String**~~ *Internal-String, and External-String*.

Change clauses 4.3.16 through 4.3.18 to read:

> **4.3.16 ~~String Value~~**String, Internal-String, and External-String Values
>
> 4.3.16.1 String Value
>
> A string value is ~~a member of the type **String** and~~ either a member of the type Internal-String or a member of the type External-String. It is a finite ordered sequence of zero or more 16-bit unsigned integer values.

NOTE: Although each value usually represents a single 16-bit unit of UTF-16 text, the language does not place any restrictions or requirements on the values other than that they be 16-bit unsigned integers.

> 4.3.16.2 Internal-String Value
>
> An internal-string value is a member of the type Internal-String. See clause 4.3.16.1 for details.

4.3.16.3 External-String Value

An external-string value is a member of the type External-String. See clause 4.3.16.1 for details.

4.3.17 ~~String Type~~ String, Internal-String, and External-String Types

4.3.17.1 String Type

The type **String** is the set of all string values.

4.3.17.2 Internal-String Type

The type Internal-String is the set of all internal-string values.

4.3.17.3 External-String Type

The type External-String is the set of all external-string values.

In section 4.8.4, change the second paragraph after the BNF to read:

A string literal stands for a value of the ~~String~~ *Internal-String* type. The ~~string~~ *internal-string* value (SV) of the literal is described in terms of the character values (CV) contributed by the various parts of the string literal. As part of this process, some characters within the string literal are interpreted as having a Mathematical Value (MV), as described below or in section 7.8.3.

Change the first line of section 8.4 to read:

8.4 The ~~String Type~~ Internal-String and External-String Types

Insert the following immediately before section 8.5:

8.4.1 The Internal-String Type

The Internal-String type is used for strings that originate purely from string literals within an ECMAScript program, from conversions of non-external-string types to a string type, and from the methods of built-in objects other than String object. The methods of the String object will return either Internal-String or External-String values, depending on the type of the string used to initialize the String object.

8.4.2 The External-String Type

The External-String type is used for strings that are returned from host objects.

In section 9.1, change the table row for Input Type String into the following two table rows:

~~String~~*Internal-String*	The result equals the input argument (no conversion).
External-String	The result equals the input argument (no conversion).

In section 9.2, change the table row for Input Type String into the following two table rows:

~~String~~*Internal-String*	The result is **false** if the argument is the empty string (its length is zero); otherwise the result is **true**.
External-String	The result is false if the argument is the empty string (its length is zero); otherwise the result is true.

In section 9.3, change the table row for Input Type String into the following two table rows:

~~String~~*Internal-String*	See grammar and note below.
External-String	See grammar and note below.

Change the first line of section 9.3.1 to read:

9.3.1 ToNumber Applied to the ~~String Type~~ Internal-String and External-String Types

In section 9.8, change the table to read:

Input Type	Result.
Undefined	*The Internal-String value* **"undefined".**
Null	*The Internal-String value* **"null".**
Boolean	If the argument is **true**, then the result is *the Internal-String value* **"true"**. If the argument is **false**, then the result is *the Internal-String value* **"false".**
Number	See note below.
~~String~~*Internal-String*	Return the input argument (no conversion).
External-String	Return the input argument (no conversion).
Object	Apply the following steps: Call ToPrimitive(input argument, hint String). Call ToString(Result(1)). Return Result(2).

Change the first paragraph under section 9.8.1 to read:

The operator ToString converts a number m to ~~string format~~ *an internal-string value* as follows:

Immediately before section 10, insert:

9.10 ToInternalString

The operator ToInternalString converts its argument to a value of type Internal-String according to the following table:

Input Type	Result.
Undefined	the Internal-String value "undefined".
Null	The Internal-String value "null".
Boolean	If the argument is true, then the result is the Internal-String value "true". If the argument is false, then the result is the Internal-String value "false".
Number	See note in section 9.8.
Internal-String	Return the input argument (no conversion).
External-String	if CanConvertToInternal(argument) returns true, then the result is an internal string with the same sequence of (16-bit) character values. Otherwise, throw SecurityError.
Object	Apply the following steps: Call ToPrimitive(input argument, hint String). Call ToString(Result(1)). If CanConvertToInternal(Result(2)) returns true, return Result(2). Otherwise, throw SecurityError.

The operator CanConvertToInternal is implemented so as to return false if the ECMAScript is being executed in a trusted context and no permissions have been granted to the context to convert external strings to internal strings. In all other cases, the operator CanConvertToInternal will return true.

Change section 10.1.2 to read:

There are three types of ECMAScript executable code:

- Global code is source text that is treated as an ECMAScript Program. The global code of a particular Program does not include any source text that is parsed as part of a FunctionBody.

- Eval code is the source text supplied to the built-in **eval** function. More precisely, if the parameter to the built-in **eval** function is ~~a string~~ *an internal-string*, it is treated as an ECMAScript Program. The eval code for a particular invocation of **eval** is the global code portion of the ~~string~~ *internal-string* parameter.

- Function code is source text that is parsed as part of a FunctionBody. The function code of a particular FunctionBody does not include any source text that is parsed as part of a nested FunctionBody. Function code also denotes the source text supplied when using the built-in **Function** object as a constructor. More precisely, the last parameter provided to the **Function** constructor is converted to ~~a string~~ *an internal-string* and treated as the FunctionBody. If more than one parameter is provided to the **Function** constructor, all parameters except the last one are converted to strings and concatenated together, separated by commas. The resulting string is interpreted as the FormalParameterList for the FunctionBody defined by the last parameter. The function code for a particular instantiation of a **Function** does not include any source text that is parsed as part of a nested FunctionBody.

In section 11.4.3, change the single table row for Type String into the following two table rows:

~~String~~ *Internal-String* "string".

External-String "string".

In section 11.6.1, change steps 14 and 15 to read:

~~14. Concatenate Result(12) followed by Result(13).~~
~~15. Result Result(14)~~
14. If Result(12) is of type internal-string and Result(13) is of type external-string, use a string value with the same sequence of characters as Result(12) but of type external-string. Otherwise use Result(12).
15. If Result(12) is of type external-string and Result(13) is of type internal-string, use a string value with the same sequence of characters as Result(13) but of type external-string. Otherwise use Result(13).
16. Concatenate Result(14) followed by Result(15). The string type is the same as that of Result(14).
17. Return Result(16).

Append the following paragraph at the end of section 12.6.4:

Property names are maintained as internal-strings or external-strings depending on the source of the property name. Property names are returned as internal-strings unless the property name was set from an external string, in which case it is returned as an external string.

Change step 1 under section 15.1.2.1 to read:

1. If x is not a string value, return x. If x is an external-string value and CanConvertToInternal(x) returns false, throw the exception SecurityError.

In section 15.1.3, change step 4 in the procedure for the hidden function Encode to read:

4. If k equals Result(1), return ~~R~~ a string with the same sequences of characters as R and the same (internal-string vs external-string) type as the input string.

In section 15.1.3, change step 4 in the procedure for the hidden function Decode to read:

4. If k equals Result(1), return ~~R~~ a string with the same sequences of characters as R and the same (internal-string vs external-string) type as the input string.

In section 15.2.4.2, change step 2 to read:

2. Compute ~~a string~~ *an internal-string* value by concatenating the three strings **"[object "**, Result(1), and **"]"**.

In section 15.3.2.1, change step 13 to read:

13. Call ~~ToString~~ *ToInternalString*(body).

In section 15.4.4.3, change step 14 to read:

Let R be a string value produced by concatenating S and Result(13). The type of R is external-string if the type of either S or Result(13) is external-string. Otherwise, the type of R is internal-string.

In section 15.4.4.5, change step 11 to read:

> Let S be a string value produced by concatenating R and Result(4). The type of S is external-string if the type of either R or Result(4) is external-string. Otherwise, the type of S is internal-string.

In section 15.4.4.5, change step 14 to read:

> Let R be a string value produced by concatenating S and Result(13). The type of R is external-string if the type of either S or Result(13) is external-string. Otherwise, the type of R is internal-string.

Change the first sentence of section 15.5.3.2 to read:

> Return ~~a string~~ *an external-string* value containing as many characters as the number of arguments.

Add the following sentence to the end of the first paragraph in section 15.5.4.2 and to the first paragraph in section 15.5.4.3:

> Return the value as an external-string if the String object was constructed with a value that converted to an external-string. Return the value as an internal-string if the String object was constructed with a value that converted to an internal-string.

Add the following sentence after the first paragraph of section 15.5.4.4 to read:

> Returns an external-string, if the type of the result of ToString is external-string. Otherwise, the returned string is an internal-string.

Change step 5 in section 15.5.4.6 to read:

> Let R be the string value consisting of the characters in the previous value of R followed by the characters Result(4). If either the previous value of R or Result(4) is of type external-string, give R type external-string. Otherwise give R type internal-string.

Insert the following paragraph just before the note in section 15.5.4.1:

> If either the result of converting this to a string is of type external-string or the replaceValue converted to a string is of type external-string or the function replaceValue returns a value of type external-string, then the newstring has type external-string. In all other cases, the newstring has type internal-string.

In section 15.5.4.13 to read, change step 8 to read:

> 8. Return a string containing Result(7) consecutive characters from Result(1) beginning with the character at position Result(5). *The type of the returned string is external-string if the type of Result(1) is external-string. The type of the returned string is internal-string if the type of Result(1) is internal-string.*

In section 15.5.4.14, change step 16 to read:

> 16. Let T be a string value equal to the substring of S consisting of the characters at positions p (inclusive) through q (exclusive). *Let the type of T (internal-string versus external-string) be the same as the type of S.*

In section 15.5.4.14, change step 28 to read:

> 28. Let T be a string value equal to the substring of S consisting of the characters at positions p (inclusive) through q (exclusive). *Let the type of T (internal-string versus external-string) be the same as the type of S.*

In section 15.5.4.15, append the following to the end of step 9:

> Let the type of the returned string (internal-string versus external-string) be the same as that of Result(1).

In section 15.5.4.16, insert the following paragraph before note 1:

> The result string's type (internal-string vs external-string) is the same as that of the string computed by converting the object to a string.

In sections 15.9.5.2 through 15.9.5.7, change the first sentence of the first paragraph in each section to read:

> This function returns ~~a string~~ *an internal-string* value.

In section 15.10.6.2, append the following at the end section:

> The type of the resulting string (internal-string vs external-string) is the same as that of the string argument.

In section 15.10.6.4, append the following at the end of the second paragraph:

> The returned string is of type internal-string if source is an internal-string. The returned string is of type external-string if source is an external-string.

Insert the following immediately after section 15.1.6.6:

> 15.1.6.7 SecurityError
>
> Indicates that an authorized attempt was made to convert an external-string into an internal-string within the context of trusted ECMAScript.

8.14.2.3 Sources of Unsafe (external) strings

8.14.2.3.1 Sources within ECMAScript

As specified in the previous section the following sources of unsafe strings within the language itself shall be typed as external strings:

a) Strings resulting from operations on external-strings.

b) `String.fromCharCode`, which converts a character code into a string.

8.14.2.3.2 Sources from Host Objects

All strings returned by Host Object APIs defined as part of the present document shall be typed as external strings.

All strings returned by the Bridge shall be typed as external strings.

Any Host Object APIs provided by a platform beyond those required in the present document should return external strings unless the result is known to be secure.

8.14.2.4 Use of strings in RCEs

An attempt to use an external string or a String Object containing an external string in a runtime code extension shall cause the DVBException with the error code SECURITY_VIOLATION_ERR to be raised.

An attempt to use an internal string or a String Object containing an internal string in a runtime code extension shall not cause the DVBException with the error code SECURITY_VIOLATION_ERR to be raised.

If the permission request file requests the permission to do privileged runtime code extension and this is granted for external strings then use of an external string or a String Object containing an external string in a runtime code extension shall be considered use of an internal string or a String Object containing an internal string.

If no permission to do privileged runtime code extension is granted then any attempt to use an internal or external string or a String Object containing an internal or external string in a runtime code extension shall be considered use of an external string.

If the permission request file requests the permission to do privileged runtime code extension and this is granted for internal strings only then use of external and internal strings retain their type.

The following uses of a string shall be considered runtime code extensions:

A string:

a) Passed to `eval()`.

b) Passed to the constructor of a `Function()` object.

c) Passed to `document.write()`.

d) Used to set an event handler.

The exception occurs when an external-string is associated with the event handler either by setting a handler attribute of an element in the document or by parenting.

e) Used as the code in a script element.

The setting of a <script> element text or src attribute in a displayed document shall not cause the new text to be incorporated into the ECMAScript context for the document. Therefore this is not strictly runtime code exception. However modification of script nodes is disallowed in the present document. The exception occurs when the external-string is associated with the script element either by parenting to a script element a node containing an external-string (e.g. by `Node::insertBefore()`, `Node::appendChild()`, `Node::replaceChild()`) or by setting the nodeValue of a TextNode or a CDATASection child of a script element to an external string.

f) Used in window.setTimeout().

8.14.2.5 Mutation of Host Objects

Any attempt to remove or replace the pre-defined properties of a host object shall cause the DVBException with the error code SECURITY_VIOLATION_ERR to be raised.

Any attempt to remove or replace the pre-defined function property of a host object shall cause the DVBException with the error code SECURITY_VIOLATION_ERR to be raised.

8.14.3 Inter-application security

8.14.3.1 Restrictions on DOM elements introduced for security

No access is provided to documents from other applications.

The property accessor on the window and document host objects shall restrict access to properties and functions of those objects when the document context in which the access is executed is from a different DVB-HTML application or from a different domain than the document in the window:

- For such contexts, the window accessor shall only allow the assignment of a string to the location property: setting any other window property or getting any window property shall throw an exception.
- For such contexts, the document accessor shall not allow any access: setting or getting any document property shall throw an exception.

The exception that is thrown shall be the appropriate exception as if an attempt to access a resource for which the caller did not have permission were made:

- In the Java binding (see clause AD.1, "Java bindings to DVB extensions") a SecurityException shall be thrown.
- In the ECMAScript binding (see clause AC, "ECMAScript Binding") when a security exception is raised, the DVBException with the error code SECURITY_VIOLATION_ERR is raised.

If a new application is started in a frame, then a new Window object is created to represent the frame. The parent of this window is the window that represents the frame from the calling applications perspective. This parent property however is not visible to the application within the frame. Therefore both applications have a window object that controls the frame, but no communication is possible between them. Access to properties, methods, and variables involving Window objects are restricted to windows that are displaying documents from the same application and from the same domain (see clause 8.16.6, "Domain").

Where an application within a frame terminates prior to the application which contains the frameset that hosts the frame for that application, then the latter application shall receive the DVB lifecycle event "AppTerminating" (see clause 8.11.2.2, "Lifecycle events").

The user agent shall set the content of the frame that contained the terminating application to a blank document. and the URL and Location property shall be set to the null string prior to sending the "AppTerminating" event.

Where an outer application terminates with a frameset which contains one or more embedded applications in frames, then the applications in those frames shall also be terminated. This shall apply even in the case where the application within the sub frame would survive service selection but the outer application will not.

8.15 DVB-HTML permissions

This section explains how the permissions that define the sandbox and are available to unsigned applications and the permissions that can be requested in the permission request file are mapped to the DVB-HTML application model. ADVB-HTML application has the following ways of accessing beyond the sandbox.

a) Following embedded links in the page can cause the loading of new resources.

b) Script may access APIs over the Java Bridge.

c) Properties in the host objects available to script may access resources.

   The location attribute of the Window object.

   The `open()` Function attribute of the Window object.

d) url() references in CSS attributes.

In case c) setting of attributes via the DOM interfaces is handled only when the attribute is used, and in this case the restrictions of case a) are applied.

In some cases a permission is only relevant to APIs reached via the Java Bridge.

8.15.1 Permissions for unsigned applications

The MHP security policy includes a set of resources that are always guaranteed to be granted to applications, if the application is executed. Unsigned applications have access to only these resources.

This section defines the mapping of those resources to DVB-HTML applications. DVB-HTML applications shall always be granted the necessary Permission.

8.15.1.1 java.awt.AWTPermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.1.1, "java.awt.AWTPermission".

8.15.1.2 java.net.SocketPermission:

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.1.2, "java.net.SocketPermission".

8.15.1.3 java.util.PropertyPermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.1.3, "java.util.PropertyPermission".

8.15.1.4 java.lang.RuntimePermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.1.4, "java.lang.RuntimePermission".

8.15.1.5 java.io.SerializablePermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.1.5, "java.io.SerializablePermission".

8.15.1.6 java.io.FilePermission

Permission to read resources for the sub-tree under which the implementation mounts the object carousels shall be granted. For APIs accessed via the, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.1.6, "java.io.FilePermission".

The location attribute of the Window object may be set to a resource in the sub-tree under which the implementation mounts the object carousels using a URL with the "dvb:" protocol.

The `open()` Function attribute of the Window object may be called with reference to a resource in the sub-tree under which the implementation mounts the object carousels using a URL with the "dvb:" protocol.

DVB HTML can read resources for the sub-tree under which the implementation mounts the object carousels using the dvb: protocol in a URL in the following element attributes:

- a.href.
- area.href.
- base.href.
- blockquote.cite.
- frame.longdesc.
- frame.src.
- iframe.longdesc.
- iframe.src.
- img.src.
- img.longdesc.
- img.usemap.
- input.src.
- input.usemap.
- link.href.
- object.archive.
- object.classid.
- object.codebase.
- object.data.
- object.usemap.
- q.cite.
- script.src.
- Using the CSS url() value.

Using a "dvb:" locator that references a resource in the Carousel on other element attributes is undefined.

8.15.1.7 javax.tv.media.MediaSelectPermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.1.7, "javax.tv.media.MediaSelectPermission".

8.15.1.8 javax.tv.service.ReadPermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.1.8, "javax.tv.service.ReadPermission".

8.15.1.9 javax.tv.service.selection.ServiceContextPermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.1.9, "javax.tv.service.selection.ServiceContextPermission".

8.15.1.10 java.util.Locale.setDefault

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.1.10, "java.util.Locale.setDefault".

8.15.1.11 org.dvb.security.PrivilegedRCEPermission

Unsigned applications are granted PrivilegedRCEPermission("external"), see clause 8.14.2.4, "Use of strings in RCEs".

8.15.2 Additional Permissions for signed applications

Signed applications can additionally request to get more permissions. These permissions are requested using the permission request file (clause 12.6, "Security policy for applications"). This section defines the mapping from the items in the permission request file to the permissions that may be granted by the MHP terminal in response to the request. And how this affects the DVB-HTML application.

In the case of org.dvb.security.PrivilegedRCEPermission only, signed applications with a permission request file can request permission that is more restrictive than that granted to applications without a permission request file.

8.15.2.1 java.util.PropertyPermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.2.1, "java.util.PropertyPermission".

8.15.2.2 java.io.FilePermission

A signed application has the rights of an unsigned application, in addition when the permission request file requests the permission to access persistent storage and this is granted, read and write access to the persistent storage directory sub-tree is allowed.

This Permission is relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.2.2, "java.io.FilePermission".

If the permission is granted, the location attribute of the Window object may be set to a resource in the persistent file store using a URL with the "file:" protocol. If the permission is not granted then attempting to set this attribute with such a locator will cause a SecurityException to be thrown.

f the permission is granted, the `open()` Function attribute of the Window object may be called with reference to a resource in the persistent file store using a URL with the "file:" protocol. If the permission is not granted then calling `open()` with such a locator will cause a SecurityException to be thrown.

DVB HTML can read the persistent file store using the file: protocol in a URL in the following element attributes:

- a.href.
- area.href.
- base.href.
- blockquote.cite.

- frame.longdesc.
- frame.src.
- iframe.longdesc.
- iframe.src.
- img.src.
- img.longdesc.
- img.usemap.
- input.src.
- input.usemap.
- link.href.
- object.archive.
- object.classid.
- object.codebase.
- object.data.
- object.usemap.
- q.cite.
- script.src.
- Using the CSS url() value.

If the file permission is not granted, all "file:" references shall fail as if the referenced resource was not available.

Using a "file:" locator that references a resource in the persistent store on other element attributes is undefined.

When there is a persistent file credential in the permission request file and this is granted, other files are made available as follows:

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

8.15.2.3 org.dvb.net.ca.CAPermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.2.3, "org.dvb.net.ca.CAPermission".

8.15.2.4 org.dvb.application.AppsControlPermission

This Permission is relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.2.4, "org.dvb.application.AppsControlPermission".

If the permission is granted, the location attribute of the Window object may be set to launch a DVB-HTML application in a sub-window by using a locator which references another DVB-HTML application. If the permission is not granted or the locator points to an application of a type which is not embeddable in a frame then attempting to set this attribute with such a locator will cause a SecurityException to be thrown.

If the permission is granted, the `open()` Function attribute of the Window object may be called with reference to a resource that launches a DVB-HTML application in a sub-window by using a locator which references another DVB-HTML application. If the permission is not granted or the locator points to an application of a type which is not embeddable in a frame then calling the `open()` with such a locator will cause a SecurityException to be thrown.

DVB-HTML applications can, if granted this permission, launch applications using locators that reference other applications (see clause 9.3.1.4, "Application boundary" and GEM [1], clause 9.2.1, "Starting DVB-J Applications") using the attributes on elements below:

- a.href.
- area.href.

If the application is not granted the permission then these links will fail as if the resource was not available.

Using a locator that references an application on other element attributes is not required to be supported.

8.15.2.5 org.dvb.net.rc.RCPermission

This Permission is relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.2.5, "org.dvb.net.rc.RCPermission".

If the permission is granted for the default ISP, setting the location attribute of the Window object using locators that would require creating a return channel link may establish the link required to access the resource. If the permission is not granted then attempting to set this attribute with such a locator will cause a SecurityException to be thrown.

If the permission is granted for the default ISP, using the `open()` Function attribute of the Window object using locators that would require creating a return channel link may establish the link required to access the resource If the permission is not granted then calling `open()` with such a locator will cause a SecurityException to be thrown.

DVB-HTML applications can, if granted this permission for the default ISP establish the link required to reference resources in any element attribute that allows references when using locators that would require creating a return channel link.

If the permission is not granted, the link will not be established and the link will fail as if the resource did not exist.

8.15.2.6 org.dvb.net.tuning.TunerPermission

This Permission is relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.2.6, "org.dvb.net.tuning.TunerPermission".

Setting the location attribute of the Window object using locators that reference an audio or video stream will cause a SecurityException to be thrown.

Calling the `open()` Function attribute of the Window object using locators that reference an audio or video stream will cause a SecurityException to be thrown.

DVB-HTML applications can, if granted this permission, reference audio and video resources that require tuning using `dvb:` locators on the following element attributes:

- img.src.
- input.src.
- object.data.
- Using the CSS url() value.

If the permission is not granted, no tuning will occur and the link will fail as if the resource did not exist.

Using a "dvb:" locator that references an audio or video resources on other element attributes is not required to be supported.

8.15.2.7 javax.tv.service.selection.SelectPermission

This Permission is relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.2.7 "javax.tv.service.selection.SelectPermission".

Setting the location attribute of the Window object using locators that references a DVB service will cause a SecurityException to be thrown.

Calling the `open()` Function attribute of the Window object using locators that references a DVB service will cause a SecurityException to be thrown.

Applications can, if granted this permission, select away from the current service to a new service using `dvb:locators` that reference a DVB service on the following element attributes:

- a.href.
- area.href.

If the permission is not granted, no service selection will occur and the link will fail as if the resource did not exist.

Using a "dvb:" locator that references a DVB service on other element attributes is not required to be supported.

8.15.2.8 org.dvb.user.UserPreferencePermission

This Permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.2.8, "org.dvb.user.UserPreferencePermission".

8.15.2.9 java.net.SocketPermission

This Permission is relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.2.9, "java.net.SocketPermission".

If the permission is granted (and if necessary the rc permission), the location attribute of the Window object may be set to a locator that references a resource on the host named in the permission file. If the permission is not granted then attempting to set this attribute with such a locator will cause a SecurityException to be thrown.

If the permission is granted (and if necessary the rc permission), the `open()` Function attribute of the Window object may be called with locator that references a resource on the host named in the permission file. If the permission is not granted then calling `open()` with such a locator will cause a SecurityException to be thrown.

DVB-HTML applications can, if granted this permission (and if necessary the rc permission), reference resources in any element attribute that allows such references using locators to resources on the host named in the permission file.

If the permission is not granted the link will fail as if the resource did not exist.

8.15.2.10 org.dvb.media.DripFeedPermission

This Permission is relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.2.10, "org.dvb.media.DripFeedPermission".

Setting the location attribute of the Window object using locators that reference a dripfeed resource will cause a SecurityException to be thrown.

Calling the `open()` Function attribute of the Window object using locators that reference dripfeed resources will cause a SecurityException to be thrown.

DVB-HTML applications can, if granted this permission, reference dripfeed resources using dripfeed: locators on the following element attributes:

- img.src.
- input.src.
- object.data.

- Using the CSS url() value.

If the permission is not granted, the link will fail as if the resource did not exist.

Using a "dripfeed:" locator on other element attributes is not required to be supported.

8.15.2.11 org.dvb.security.PrivilegedRCEPermission

When the permission request file requests the permission to do privileged runtime code extension on internal strings and this is granted, a PrivilegedRCEPermission with the name "internal" is created.

When the permission request file requests the permission to do privileged runtime code extension on external strings, but it is granted only for internal strings, a PrivilegedRCEPermission with the name "internal" is created.

When the permission request file requests the permission to do privileged runtime code extension on external strings and this is granted, a PrivilegedRCEPermission with the name "external" is created.

For behaviour, see clause 8.14.2.4, "Use of strings in RCEs".

See GEM [1], clause 11.10.2.12, "javax.microedition.apdu.APDUPermission".

8.15.2.12 org.dvb.application.storage.ApplicationStoragePermission

This permission is only relevant to API use over the bridge, the same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.2.11, "org.dvb.application.storage.ApplicationStoragePermission".

8.15.2.13 javax.microedition.apdu.APDUPermission

This permission is relevant only to API use over the bridge. The same rights are granted as for DVB-J applications, see GEM [1], clause 11.10.2.12, "javax.microedition.apdu.APDUPermission".

8.16 Miscellaneous

8.16.1 Date Values

8.16.1.1 Syntax

Absolute Date values have the following syntax: [to be provided]

8.16.2 Clock values

8.16.2.1 Syntax

Absolute Clock values have the following syntax:

```
Clock-val         ::= ( Full-clock-val | Partial-clock-val | Timecount-val )
Full-clock-val    ::= Hours ":" Minutes ":" Seconds ("." Fraction)?
Partial-clock-val ::= Minutes ":" Seconds ("." Fraction)?
Timecount-val     ::= Timecount ("." Fraction)? (Metric)?
Metric            ::= "h" | "min" | "s" | "ms"
Hours             ::= DIGIT+; any positive number
Minutes           ::= 2DIGIT; range from 00 to 59
Seconds           ::= 2DIGIT; range from 00 to 59
Fraction          ::= DIGIT+
Timecount         ::= DIGIT+
2DIGIT            ::= DIGIT DIGIT
DIGIT             ::= [0-9]
```

The default metric suffix is "s" (for seconds). No embedded white space is allowed in clock values, although leading and trailing white space characters will be ignored.

The following are examples of legal clock values.

Full clock values:

```
02:30:03    = 2 hours, 30 minutes and 3 seconds
50:00:10.25 = 50 hours, 10 seconds and 250 milliseconds
```

Partial clock value:

```
02:33   = 2 minutes and 33 seconds
00:10.5 = 10.5 seconds = 10 seconds and 500 milliseconds
```

Timecount values:

```
3.2h    = 3.2 hours = 3 hours and 12 minutes
45min   = 45 minutes
30s     = 30 seconds
5ms     = 5 milliseconds
12.467  = 12 seconds and 467 milliseconds
```

Fractional values are just (base 10) floating point definitions of seconds. The number of digits allowed is unlimited (although actual precision may vary among implementations).

EXAMPLE:

```
00.5s = 500 milliseconds
00:00.005 = 5 milliseconds
```

8.16.2.2 Offset values

Offset values are used to specify when an element should begin or end relative to another time.

An offset value has the following syntax:

```
offset-value ::= "+" ( Clock-value )
```

An offset value includes a sign on a clock value, and is used to indicate a positive offset. The offset is measured from the implicit start time.

8.16.3 Unrealizable locators

Locators may not be realizable for several reasons:

- The locator requires tuning for which permission or resources are not available.
- The locator requires network or interaction channel access for which permission is not available.
- The locator requires access to data that is denied by Conditional Access mechanisms.
- The locator requires access to a authenticated file for which authentication fails.
- The locator requires mounting a file system that cannot be mounted at the same time as the currently mounted file systems.

8.16.3.1 Presentation of Locators in DVB HTML

It is implementation specific how the user agent indicates the presence of links to resources when rendering DVB-HTML. Where the user agent can determine a priori that a given link will fail, it is implementation dependent whether the user agent renders this as a link. It is also implementation dependent if and how a user agent presents failure to locate a resource due to insufficient permission being granted.

NOTE: HTML 4 [41], section 12.2.4 recommends that the user agent alerts the user when a referenced file is not available. To avoid computer-like dialogs and accessibility issues it is recommended that authors follow the guidelines in Accessibility e.g. providing text equivalents using the `alt` attribute.

8.16.4 Relation to HTTP and HTTPS

The user agent string (see clause 8.13.1, "User agent strings") is sent in a header as part of all HTTP and HTTPS requests (see clause 6.3.7, "HyperText Transfer Protocol (HTTP)").

8.16.5 DVB-HTML specific locators

DVB-HTML has two specific formats of locators which have defined semantics only in the context of DVB-HTML:

- The extended form of the DVB locator (see clause 8.16.5.1, "Extended DVB locator").
- The exit locator used for application self termination (see clause 8.16.5.2, "Exit locator").

8.16.5.1 Extended DVB locator

8.16.5.1.1 Extended DVB locator syntax

The formal specification of the URL form expressed in BNF is given in the following extension to the dvb: locators defined in table 101, "DVB URL syntax".

Table 61: Extended DVB URL syntax

```
dvbhtml_url                          = dvb_url | dvb_scheme ":" dvbhtml_hierpart
dvbhtml_hierpart                   = dvbhtml_net_path
dvbhtml_net_path                   = "//" dvbhtml_entity
dvbhtml_entity                       = dvb_service_contextual | dvb_service_component_contextual
| ait_specifier
dvb_service_contextual             = "current" | "original"
dvb_service_component_contextual     = "current.audio" | "current.video" | "current.av"
ait_specifier                      = ait_filter "." "ait" ait_abs_path
ait_filter                         = "current"
ait_abs_path                         = "/" ait_entity
ait_entity                         = ait_root_directory | ait_application
ait_root_directory                   = "app_root"
ait_application                      = org_id_part "." app_id_part [ "?" ait_params ]
ait_params                         = "arg_" 1*digit "=" *uric ["&" ait_params ]
```

NOTE 1: For `digit` and `uric` see RFC 2396 [18].
NOTE 2: For `org_id_part` and `app_id_part` see clause 14.5, "Text encoding of application identifiers".

The semantics of launching applications via such a locator are specified in clause 8.5.3.1.3, "Hypertext".

8.16.5.1.2 TV locators

A locator for a DVB service or service component can be a full dvb: locator, as defined in clause 14.1, "Namespace mapping (DVB Locator)", or one of the following specific forms.

Table 62

Locator	Meaning
dvb://current	The service currently selected by the application.
dvb://current.av	The Audio and Video being presented on the background video device (see clause 11.6.2, "Service Selection API").
dvb://current.audio	The Audio being presented in association with the background video device.
dvb://current.video	The Video being presented on the background video device.
dvb://original	Originating service for this application (place of birth).
NOTE: Content authors who, in other systems, use the "tv:" locator, as defined in RFC 2838 [53], may use the equivalent "dvb://current.av" locator to reference the default audio and video component within the service.	

8.16.5.1.3 Application locator

See clause 14.1.7, "Application locator".

8.16.5.1.4 AIT locators

Locators for the root directory or the icon representation of the current application can be identified by the following specific forms.

Table 63

Locator	Meaning
dvb://current.ait/app_root	The root directory path as found in the dvb html application location descriptor for the application (see note).
dvb://current.ait/app_icon	The Icon found in the application icons descriptor for the application (see note).
NOTE: Shall mount file systems if referenced objects are in unmounted file systems and the link is followed.	

8.16.5.2 Exit locator

In the context of a DVB-HTML application, actioning a link in the defined element, attribute context (see table 9, "MIME media type usage") with the following form of locator shall cause an application to terminate.

```
exit:
```

The formal specification of the URL is given in the following BNF.

Table 64: Exit URL syntax

```
exit_url                                = exit_scheme ":" *uric
exit_scheme                             = "exit"
```

Activating such a link shall request that the application manager move the current application into the Killed state. Any possible characters following the ":" shall be ignored in the present document (see RFC 2396 [18]).

8.16.6 Domain

One of:

- The domain name of the server that served the document if it can be identified.
- An empty string ("") for a page with a dvb: locator.

NOTE: Such pages cannot therefore be in the same domain as any page delivered via http.

- Null otherwise.

9 Application model

In this clause, the term "application description" is to be interpreted as referring to the Application Information Table (AIT) defined in clause 10.4 of the present document. Additionally, attention is drawn to the general rules in clause 4.2, "Conventions".

9.1 Broadcast MHP applications

GEM [1], clause 9.1 is included in the present document, with the following notes and modifications.

9.1.1 Basic lifecycle control

GEM [1], clause 9.1.1 is included in the present document, with the following notes and modifications:

- In a DVB-HTML application, a service context is represented by a JavaObject object returned by the Bridge API, having the methods and attributes corresponding to the `ServiceContext` class. Where multiple DVB-HTML applications are being presented in the same service context, the number of JavaObject objects representing a service context is implementation dependent, but each application sees only one such instance. Changes made by one application to the JavaObject object that it has are visible to JavaObject objects representing the same service context in other applications. DVB-HTML applications may obtain a reference to the service context within which they are executing through the Bridge API to access the `getServiceContext(XletContext)` on the `ServiceContextFactory` class.

- In a DVB-HTML application, selecting a service either corresponds to activating a link to a `dvb: locator` representing a service, or by calling the `select()` method on a Java object representing a service context.

- In an internet access application, selecting a service corresponds to activating a link to a `dvb:` locator representing a service.

9.1.2 Starting applications

GEM [1], clause 9.1.2 is included in the present document, with the following notes and modifications. To the last two paragraphs (reproduced here for clarity), add the bullet point regarding the behaviour of a DVB-HTML application:

Where the currently selected service in a service context includes multiple GEM applications, any running applications may be able to launch other applications from that set. The launched applications shall be presented inside that same service context.

- *A DVB-J application is able to achieve this using the application listing and launching API.*

- A DVB-HTML application is able to achieve this either by activating a link using a locator referencing an application, or by using the Bridge APIs.

9.1.3 Support for execution of multiple simultaneous applications

GEM [1], clause 9.1.3 is included in the present document.

9.1.4 Stopping applications

GEM [1], clause 9.1.4 is included in the present document.

9.1.5 Persistence of Applications Across Service Boundaries

GEM [1], clause 9.1.5 is included in the present document, with the following notes and modifications:

If temporarily disconnected, the broadcast carousel shall be reconnected upon selection of a service where an identical carousel is available. This includes both the original service from which the application was downloaded (assuming the carousel is still present in that service) and other services containing an identical carousel as defined by clause B.2.10, "Delivery of Carousel within multiple services". To determine carousel availability, the MHP shall use the PMT information obtained for the PMT of the currently selected service.

9.1.6 Management of autostarting

GEM [1], clause 9.1.6 is included in the present document.

9.1.7 Tuning without service selection

GEM [1], clause 9.1.7 is included in the present document, with the following notes and modifications.

The reference to GEM [1], clause 10.4.2, "Visibility of Application Description and tuning" shall be understood to mean clause 10.4.4, "Visibility of AIT".

The service being presented in the service context shall not be changed by an application using these mechanisms:

- DVB-J tuning APIs.
- DVB-HTML accessing a media or service reference (e.g in an <object> or <img> element), or by accessing the tuning API through the Bridge.

9.1.8 MHP Applications and Service Selection

GEM [1], clause 9.1.8 is included in the present document.

9.1.9 Cached applications

GEM [1], clause 9.1.9 is included in the present document.

9.2 DVB-J Model

GEM [1], clause 9.2 is included in the present document, with the following notes and modifications:

In GEM [1], clause 9.2.3.2, the reference to the `application_control_code` parameter of the application description is to be interpreted as referring to the `application_control_code` defined in clause 10.4, "Application Information Table".

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 present document. The extent of the set is described by the application boundary.

9.3.1.2 User agent

A user agent is an application that interprets a content format (in this case DVB-HTML documents).

NOTE: This could be implemented as a plug-in.

9.3.1.3 DVB-HTML Actor

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

There is a single DVB-HTML Actor for each running DVB-HTML Application, each DVB-HTML Application can consist of multiple documents several of which could be simultaneously displayed.

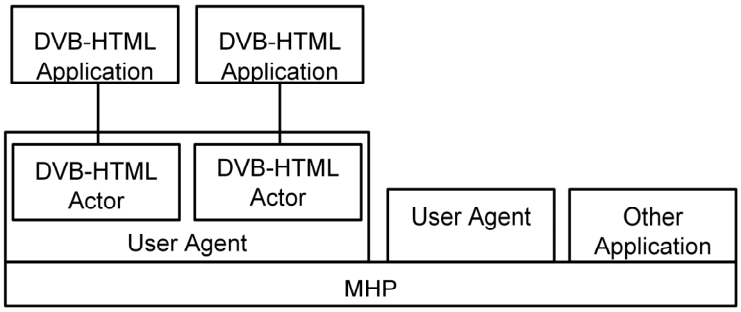

Figure 9: Relationships between actors and applications

9.3.1.4 Application boundary

The application boundary defines the extent of a DVB-HTML Application. Any content documents outside of the application boundary are considered to not be part of the DVB-HTML Application and shall be unavailable to the referencing DVB-HTML application. An MHP terminal may contain an implementation specific mechanism to allow the end-user to decide to explicitly access the resource outside the boundary but this shall not be in the context of the original DVB-HTML application. This namespace can be used by a broadcaster to easily control the perimeter of user navigation and by an MHP implementation to more efficiently pre-fetch applications.

The logical extent of a DVB-HTML application could potentially be quite large, and for various reasons might not all be on the MHP terminal at one time. Part of it might be broadcast, part of it might be on local storage, part of it might be on the world wide web - some of it may even be generated on demand. For this reason the compact format of the regular expression is used to define the extent. The set of DVB-HTML documents making up a DVB-HTML application is defined by a regular expression over the locator language; broadly a locator consists of a text string in the following form from RFC 2396 [18] 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].

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 [54], section 2.8.4.

Relative locators in the DVB-HTML application are expanded to a full URI as defined in RFC 2396 [18], (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 clause 10.10.3, "DVB-HTML application boundary descriptor"). This label can be used for pre fetching in a transport specific manner, for example in an object carousel it defines that all modules matching the label should be preloaded.

For example: an application consists of an entry web page /phase0/index.html, and is factored into three sub sections, each of which has an associated stylesheet and image directory.

```
labelA: (/phase0/.+\.html | /phase0/images1/.+\.png | /phase0/scripts1/.+\.js)
labelB: (/phase1/.+\.html | /phase1/images/.+\.png | /phase1/scripts/.+\.js)
labelC: (/phase2/.+\.html | /phase2/images/.+\.png | /phase2/scripts/.+\.js)
```

The entry point locator signalled for this application matches the first regular expression, this allows the pre-fetch mechanism to load the modules labelled with labelA (which the broadcaster arranges to contain the contents of directory phase0), Once the user agent is running, it can use this information to detect which if any links from the current page might transition to a new phase and therefore require more pre-fetching.

9.3.2 DVB-HTML Application Lifecycle

9.3.2.1 Introduction

There are three key parts of the DVB-HTML application lifecycle model:

a) How applications are signalled as available to the MHP, and for auto start and prefetch applications how the start time is synchronized with any associated media stream.

b) How and when the application manager or other launcher application makes the presence of an non auto-start application known to the user and provides it with a trigger. This is covered by the application discovery and launching mechanisms.

c) How a broadcaster controls an actor after it has started.

9.3.2.2 Signalling

The DVB-HTML Application is signalled as described in clause 10, "Application Signalling".

The application manager can be requested to start a DVB-HTML application either because it is signalled as auto start, or through the application launching API.

On receiving the request that a DVB-HTML application is to be started (i.e. an AUTOSTART or PREFETCH appears in the AIT or it is user instantiated), and there is no application with the same applicationID already instantiated, the application manager should attempt to find a suitable user agent. It can also at this point begin pre-fetching material.

If the application manager is unable to instantiate a user agent either through lack of resources, or no suitable user agent being available then any pre-fetching can be aborted, and any trigger signal can be ignored.

It is platform dependent 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 clauses. The transitions between states are made as a result of, for example:

- A trigger, such as a request to go to a new document.
- A trigger such as the DVB-HTML application making an explicit request to change state.
- A change in the external environment i.e. the application_control_code in the AIT changes.

Since a user agent may be performing as several actors, it can be in several of these states at one time, each actor however will be labelled with a unique application ID.

A DVB-HTML application proceeds by moving between documents, while the documents remain within the DVB-HTML application boundary the DVB-HTML application continues to run normally.

Links within an DVB-HTML application normally replace the existing document, but attributes may be present on a link which cause both the new and old document to be visible at the same time.

9.3.2.3 Lifecycle control

The state model for the DVB-HTML application lifecycle control model described in this clause reflects the signalling (see clause 10.6, "Control of application life cycle") and is an abstract view of how a DVB-HTML application operates, and considers the kinds of resources that a user agent would need in order to function properly: resources concerned with output (rendering), input (event catching) and connection (the availability of the content).

The abstract model however is mostly illustrative and does not imply any resource management strategy nor is it intended to overly constrain the implementation of a user agent.

9.3.2.3.1 State diagram

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

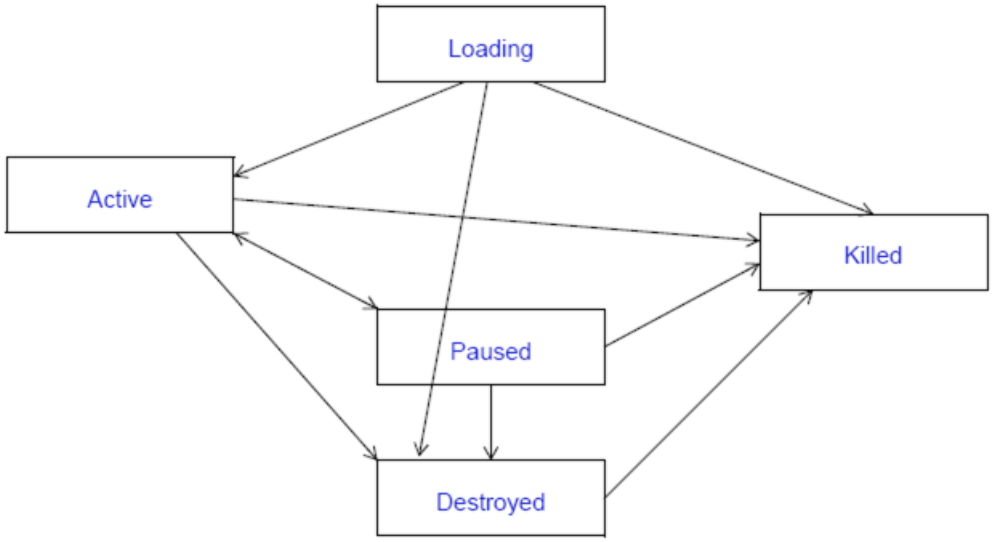

Figure 10: DVB-HTML application life cycle state diagram

9.3.3 The State Model

The entry state of the state machine, Loading, is characterized by access to the content resources and signalling resources but does not have or require input and output resources. This implies the actor can prefetch content and receive triggering events for transition to the Active state, but will not be presented to the viewer.

On entry into the Active state the actor would be assumed to have full access to the content of the present document and all resources of the MHP, subject to resource management and security issues.

The paused state is a reduced operational state. If the application manager or the user agent needs resources for other purposes, an actor may be moved to the paused state, when in this state it may no longer have full (or even any) access to resources. When the actor is reactivated it returns to its previous state.

The destroyed state can be characterized as loss of the content resource. The actor may still be able to run the DVB-HTML application due to caching or other mechanisms but must be prepared for loading of some or all of the documents from within the DVB-HTML application to fail. It is implementation dependent how such failure is handled. This is a way for the broadcaster to signal to the MHP that it is on its 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.

An actor shall not stop running itself or a DVB-HTML application without transitioning the DVB-HTML application into the killed state.

9.3.3.1 Loading

9.3.3.1.1 Name

Loading.

9.3.3.1.2 Entry actions

Instantiation of an actor.

9.3.3.1.3 Activities

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

9.3.3.1.4 Resources

Content, signalling, Output.

9.3.3.1.5 Transitions

Active. preconditions:

- Enough data is available to present something sensible.

Killed. preconditions:

- If the DVB-HTML application is signalled as KILL.

Destroyed. preconditions:

- If the DVB-HTML application is signalled as DESTROY.

9.3.3.1.6 Comment

This is the entry state of the state machine This state is entered only once in the lifetime of the DVB-HTML actor. Any start-up phase of a user agent can also be considered as part of this state. When an actor is in this state it is not rendering anything. This state should not be confused with any prefetching of modules which may be carried out by the MHP prior to application launch.

9.3.3.2 Active

9.3.3.2.1 Name

Active.

9.3.3.2.2 Activities

Gathering and parsing current document and related resources, rendering document, Maintaining rendered documents. receptive to events, waiting for triggering event to show loaded documents.

9.3.3.2.3 Entry actions

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

9.3.3.2.4 Resources

Content, signalling, output, input.

9.3.3.2.5 Transitions

Pause.

- If the user agent or application manager puts the DVB-HTML application in PAUSE.

Destroyed.

- If the DVB-HTML application is signalled as DESTROY.

Killed.

- If the DVB-HTML application is signalled as KILL.

9.3.3.2.6 Comment

This state is the steady state,

In this state it is user agent specific as to whether a partially loaded document is displayed, or deals with input triggers.

If a transition is made to a new document within the application, the actor remains in this state.

If a related resource document of the main document changes, then the resource may be reloaded, causing the DVB-HTML actor to receive appropriate DOM events, however the DVB-HTML actor is not considered to change state. Similarly the DVB-HTML actor may gain and lose the focus while in this state, receiving the appropriate DOM events - it may receive fewer input events when it does not have the focus.

If the AIT no longer refers to the DVB-HTML application no special action is taken for DVB-HTML actors that are in the Active state.

9.3.3.3 Paused

9.3.3.3.1 Name

Paused.

9.3.3.3.2 Activities

DVB-HTML actor should minimize its use of resources.

9.3.3.3.3 Resources

Application specific.

9.3.3.3.4 Transitions

Active.

- If the DVB-HTML application is resumed.

Destroyed.

- If the DVB-HTML application is signalled as DESTROY.

Killed.

- If the DVB-HTML application is signalled as KILL.

9.3.3.3.5 Comment

The semantics of this state are both user agent and DVB-HTML application specific. When the DVB-HTML application returns from the "Pause" state, the environment might have changed (loss of resources or network connections) and some events may not have been reported.

9.3.3.4 Destroyed

9.3.3.4.1 Name

Destroyed.

9.3.3.4.2 Activities

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

9.3.3.4.3 Resources

Input and output.

9.3.3.4.4 Transitions:

Killed.

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

9.3.3.4.5 Comment

This state indicates the MHP may no longer be able to access the content resources required to run the DVB-HTML application. It is DVB-HTML application and user agent specific as to whether the actor continues to run, and if it does how the user should be informed if any link is no longer available because the content it refers to is no longer available, or a cached copy has expired. The DVB-HTML actor may continue to execute in the destroyed state until the user actively dismisses it.

9.3.3.5 Killed

9.3.3.5.1 Name

Killed.

9.3.3.5.2 Entry actions

Release of resources.

9.3.3.5.3 Activities

Termination of the DVB-HTML application.

9.3.3.5.4 Resources

None.

9.3.3.5.5 Transitions

None.

9.3.3.5.6 Comment

After the activities in this state are finished, the application is no longer running. This state is the exit state of the state machine.

9.3.4 Application activity events

Each page within an application shall receive load and unload events with the following constraints:

- The DVB lifecycle `AppActive` event precedes the first DVB-HTML `load` event.
- There is no guarantee on the relative ordering of the first DVB-HTML `load` event and the DVB lifecycle `AppStarting` event.
- When a document is unloaded because of a transition to a new page the current document will receive an `unload` event and all event handlers will be allowed to complete before the new page begins its lifecycle.
- When an application transitions to the Killed state (i.e. the DVB-HTML actor ceases to execute), there is no guarantee that the current document will receive any `unload` event.
- When an application transitions to the Destroyed state the application may continue to execute in the Destroyed state, in which case `unload` events and new `load` events may be received if assets are available.
- The document `load` event is received after all parts that can receive `load` events have received them. Specify when the `load` event is fired for each media type (e.g. streamed, multipart replace).
- No non-lifecycle events are delivered to a page until after the `DVBDOMStable` event is delivered.
- Events that occurred when the application is in the from the Loading state and may be delivered when in the Active state.

NOTE: The first two constraints will not be visible to documents in frames which share an application boundary with their parent, as they do not receive DVB lifecycle events.

The following state diagram illustrates these interactions.

- DVB-HTML lifecycle states and transitions are in This Style.
- DOM events are in **This style**.
- States and transitions which are only part of this model are in *this style.*
- The *request* transition indicates a request (from user inputs, trigger events, etc.) to navigate to a new page.
- The (*no event*) transition may happen immediately, with no other input.
- The *loading page* state covers loading content for a new page and may cover full or incremental rendering of the page (although the page may not be presented to the user, if the application is signalled as PREFETCH and the AppStarting event has not been received).
- The *loading content* state covers the referenced content for a new page after the initial content is parsed and may cover full or incremental rendering of the page (although the page may not be presented to the user, if the application is signalled as PREFETCH and the AppStarting event has not been received).
- In the *loaded page* state, the page has been loaded. Unless the application is signalled as PREFETCH and no AppStarting event has occurred, it will also have begun to be presented to the user.
- In the *unloading page* state, content loading for the new page may begin.

- The AppStarting, AppPause, AppResume and AppDestroyed DVB-HTML lifecycle events are not shown in this diagram, because the only constraints on them are in relation to the standard MHP lifecycle states and events, not in relation to the DOM load and unload events.

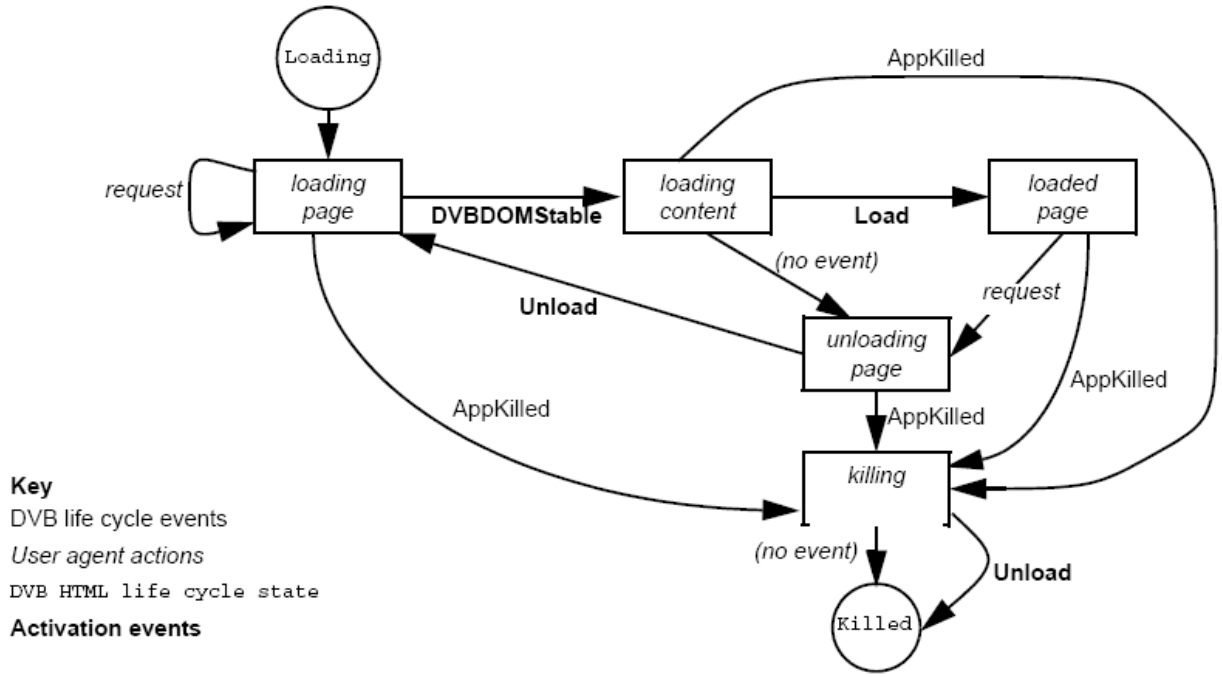

Figure 11: Activation event state model

9.3.4.1 Event queue handling

The HTML and DOM standards say little about how events are inserted by the User Agent or how they are queued. DVB-HTML User Agents shall observe the following list of clarifications.

- After the `unload` event is sent, no other events shall be sent to the document, so all trigger event listeners may be discarded. (For clarification: `AppDestroyed` and `AppKilled` happen before this.).
- When navigating to a new page in a given window or frame, the user agent may load some or all of the new page and its associated files, before sending an `unload` to the current document. It shall not begin to render the new page, or deliver DOM events to it, until event handling of the `unload` event for the old page is complete.
- In loading resources for a new page, the user agent may register for, receive and queue DSM-CC events; however it shall not deliver them to the DOM until after it has delivered the `DVBDOMStable` event.
- The ECMAScript "document" variable shall be bound to the relevant document in all event handlers.

NOTE: Together the above imply that trigger events (as is theoretically the case with user input events) may be lost during page transitions or even during presentation of a page. Authors should be aware of this possibility and code accordingly.

9.4 Inter-application resource management

GEM [1], clause 9.4 is included in the present document, with the following notes and modifications.

The reference to the `application_priority` field in the application descriptor is to be interpreted as referring to the `application_priority` defined in clause 10.4.

9.5 Life cycle of Xlets embedded in DVB-HTML

The state of an Xlet is influenced by the state of the page that contains it, in addition to being influenced by the state of the containing application. The Xlet's lifecycle is bound by the lifecycle of the HTML application and page, but the Xlet may, through its own actions, be in a less active state than the containing HTML page. For example, an Xlet may destroy itself even if the page in which it is contained is active.

9.5.1 Starting embedded Xlets

For each Xlet specified with an object tag, a new classloader shall be created to load its classes. The standard MHP semantics surrounding Xlet classloaders apply. For example, classloaders shall not be shared with other Xlets.

When an HTML page is being loaded, any Xlets embedded within that page shall be requested to enter the loaded state, as defined for the `java.xlet.Xlet interface`.

Sometime before the page becomes visible, the platform shall request all embedded Xlets in that page to enter the paused state.

During the process of making a page visible, the platform shall request all paused Xlets (i.e. those that did not fail or voluntarily terminate before reaching this state) to enter the active state. The exact point during this process at which this happens is implementation dependent, but shall not be before any load event for the page.

When the containing DVB-HTML application transitions from the paused state to the active state, all paused Xlets that are on the current page shall be requested to enter the active state.

9.5.2 Termination

When a page becomes invisible due to navigation away from that page or when the containing DVB-HTML application becomes paused, all embedded Xlets that are in the active state shall be requested to enter the paused state.

It is implementation dependent when an Xlet that was paused due to navigation is transitioned from the paused state to the destroyed state, but this shall be after any unload event for the page in which it was embedded.

When the HTML application enters the killed state, all embedded Xlets must be put into the destroyed state by the platform.

When the DVB-HTML application enters the HTML destroyed state, it has no effect on the state of any embedded Xlets. Those embedded Xlets may choose to change their state as resources are withdrawn during such a transition, however they are only forced to change state when the DVB-HTML application enters the killed state.

The standard MHP rules concerning the destruction of an Xlet apply. Notably, an Xlet that fails to respond to a lifecycle notification is eligible for destruction, even if the containing page is still visible. Similarly, an Xlet can request that it be destroyed. This will cause the Xlet to be destroyed, but it will have no effect on the lifecycle of the page in which the Xlet is contained nor shall it cause the page to be reflowed. Similarly, an Xlet that puts itself in the paused state will have no effect on the state of the enclosing HTML page or application.

Of course, an Xlet might initiate an action that causes the application containing it to be terminated. For example, if the Xlet does a service selection, the application might not be signalled in the new service. Here, the existing MHP semantics concerning application lifecycle apply.

9.5.3 General issues

Multiple Xlets per XHTML document are allowed to the same extent as multiple Xlets are allowed in the same DVB service.

Instances of embedded Xlets:

- Shall not be listed in the AIT.
- Shall not be exposed in the application listing an launching API when launched through a DVB HTML application.

- Inherit the transport protocols of the containing HTML application.

When an embedded Xlet is in its active state, it shall have DOM access. It is implementation dependent if it will have DOM access in other states.

9.6 Services and applications not related to conventional DVB services

GEM [1], clause 9.6 is included in the present document, with the following notes and modifications.

In this clause, the terms "application description" and "application descriptor" are to be interpreted as referring to the application information table (AIT) defined in clause 10.4.

In MHP, inclusion of GEM [1], clauses 9.6.3 and 9.6.4 is required.

9.7 Lifecycle of internet access applications

GEM [1], clause 9.7 is included in the present document.

9.8 Plug-ins

GEM [1], clause 9.8 is included in the present document, with the following notes and modifications.

In this clause, the terms "AIT" and "application descriptor" are to be interpreted as referring to the application information table defined in clause 10.4 of the present document.

9.9 Stored and cached applications

GEM [1], clause 9.9 is included in the present document.

9.10 Lifecycle interactions between MHP and resident applications

GEM [1] clause 9.10 "Lifecycle interactions between GEM and resident applications" is replaced with the following:

Running MHP applications shall not be killed as a direct consequence of showing the UI of a resident application (e.g. the MHP navigator). They may still be killed for reasons such as lack of resources in the receiver or as a consequence of the end-user using some kind of "application manager" user interface provided by one of the resident applications or as a result of other operations performed by the resident application that are required to kill MHP applications, e.g. channel changing by service selection.

If showing the user interface of a resident application results in the graphics of MHP applications being removed from the screen as defined in GEM [1], clause 13.3.8, the MHP applications concerned shall be put into the paused state before their graphics are removed. If other resources are automatically removed as a consequence of a resident application being started, these shall also be removed once the MHP applications concerned are in the paused state.

NOTE: This is to allow DVB-J applications to prepare for the loss of resources in their implementation of the `pauseXlet` method. For example, applications using scaled video may expand that video to full screen in their implementation of the `pauseXlet` method.

If resources that were removed when an application entered the paused state are automatically returned when that application returns to the active state, this shall be done before the `startXlet` method is called.

9.11 Providers

GEM [1], clause 9.11 is included in the present document.

9.12 Impact of graphics constraints on the application model

GEM [1], clause 9.12 is included in the present document.

9.13 Unbound applications

GEM [1], clause 9.13 is included in the present document, with the following notes and modifications.

9.13.1 Introduction to unbound applications (informative)

GEM [1], clause 9.13.1 is included in the present document.

9.13.2 Service model

GEM [1] clause 9.13.2, "Service model" is replaced by the following:

Clause 10.2.2.2 of OCAP [49], "OCAP Service Model" shall be supported.

9.13.3 Application lifecycle

GEM [1] clause 9.13.3, "Application lifecycle" is replaced by the following:

Clause 10.2.2.3 OCAP [49], "Application Signaling and Lifecycle" shall be supported.

9.13.4 Initialization of MHP Environment

GEM [1], clause 9.13.4 is included in the present document.

10 Application Signalling

10.1 Introduction

GEM [1], clause 10.1 and its clauses are included in the present document with the following additions and modifications.

This clause 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 signalling descriptor).

10.1.1 Summary of requirements on common signalling

The minimum signalling requirements for any MHP applications are summarized 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 summarized 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 summarized 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 78.

10.1.5 Summary of additional signalling for applications carried via IP

Application Information Table with the following information in its common descriptor loop:

- IP signalling descriptor.

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 79.

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 summarized below.

To add further transport protocols:

- Extend table 77, "Semantics of selector bytes".
- Possibly define further specialist descriptors such as the IP signalling descriptor.

To add further application representations:

- Define further specialist descriptors such in clause 10.9, "DVB-J specific descriptors".
- Define the application type specific life cycle control codes in clause 10.6, "Control of application life cycle".

Where constant values are registered by the present document extend the table 90, "Registry of constant values".

10.1.7 Service information

See clause 10.12, "Service Information".

10.2 Program Specific Information

GEM [1], clause 10.2 is included in the present document with the following additions and modifications.

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 (ISO/IEC 13818-1 [15] private sections).
- An Application Signalling Descriptor.

There may be more than one elementary stream carrying application signalling information for a service.

10.2.2 Data broadcast streams

The minimum signalling in the PMT associated with data broadcast components is the value of the PMT `stream_type` field required by the DVB data broadcasting specification (EN 301 192 [5]) for the transport protocol. The full details of the data broadcast protocol, the location of its "principal" component etc. are provided in the AIT (see clause 10.4, "Application Information Table").

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 clause B.2.8, "Mounting an Object Carousel".

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_identifier_descriptor (see clause B.2.8, "Mounting an Object Carousel").

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 clause 10.7.2, "Data broadcast id descriptor".

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 clause 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 clause 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. The AIT comprises the set of AIT sub-tables (see clause 10.4.5) within the selected service which have an `application_type` that the receiver can decode.

Data in the AIT allows the broadcaster to request that the receiver change the activation state of an application.

10.4.1 Data errors

AITs which contain errors shall be processed as follows:

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

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

- An error in an application information section outside of an application loop shall result in that entire application information section being silently discarded. Processing of the AIT shall continue with the next application information section (if any).

10.4.2 AIT transmission and monitoring

MHP terminals shall monitor the PMT for changes in the number of AIT elementary streams present. Changes shall be detected within 1 second. MHP terminals shall monitor all AIT elementary streams within the selected service, as described in more detail below.

The minimum repetition rate for each AIT subtable is 10 seconds. The AIT shall be carried in an elementary stream which is not scrambled.

Provided that AITs for the selected service are delivered on 3 or fewer elementary streams then the maximum time interval between the moment the AIT is updated and the moment the new version is detected by the MHP shall be no more than 30 seconds.

NOTE: If broadcasts use more than 3 elementary streams to deliver AITs then receiver response time may degrade in an unpredictable way.

The MHP terminal is only required to monitor AIT sections for application types that it can decode. The application types a terminal can decode include those types for which the terminal has available interoperable plug-ins (see GEM [1], figure 9 "Illustrative plug-in implementation options"). Available inter-operable plug-ins shall include those MHP applications of application types it can already decode which are signalled with plug-in descriptors (see clause 10.13.4, "Plug-in descriptor").

The set of application types listed in the application database reflects the set of AIT sections being monitored. So, this may be a subset of the application types being broadcast in the case that the broadcast carries a superset of the terminal's capabilities.

Applications removed from the AIT sub-table which was signalling them but where that AIT sub-table remains present in the network, shall be stopped as if they had been signalled with a `DESTROY` control code.

If the AIT sub-table signalling an application vanishes from the network completely, that application shall continue to run. The MHP terminal shall monitor for the re-appearance of the AIT sub-table as defined for the appearance of new AIT sub-tables above.

10.4.3 Optimized AIT signalling

The optional `AIT_version_number` carried by the Application Signalling Descriptor allows a possible optimization 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 clause 10.7.1, "Application signalling descriptor".

10.4.4 Visibility of AIT

If an application tunes away from a transport stream where its signalling is carried without selecting a new service, it will continue running although the AIT is not visible.

In MHP terminals with multiple network interfaces, if the AIT of the selected service is visible via any of them, then the AIT signalling is used as normal.

10.4.5 Definition of sub-table for the AIT

All sections on the same PID with the AIT `table_id` and the same value of `application_type` are members of the same sub-table.

10.4.6 Syntax of the AIT

The Application Information Section describes applications and their associated information. Each Application Information Section includes one "common" descriptor loop at the top level for descriptors that are shared between applications of that sub-table and a loop of applications. Each application in the application loop has an "application" descriptor loop containing the descriptors associated with that application.

Like DVB SI tables, the scope of common loop descriptors is the sub-table. So, any descriptors present in the common descriptor loop apply to all sections of the sub-table. Typically, common descriptors would normally only be present in section 0 of a sub-table, unless there was not enough space.

Like other DVB SI tables, any strings contained in these tables shall not have null terminations.

Table 65: 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
test_application_flag	1	bslbf
application_type	15	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 1 021 (0x3FD).

test_application_flag: This 1-bit field when set indicates an application which is transmitted for the purposes of receiver testing and which shall not be started or listed in any API or displayed in any user interface by MHP receivers under normal operational conditions. The means (if any) by which an MHP receiver is put into a mode where applications signalled with this bit set are treated as if this field is set to zero is implementation dependent but should not be one which typical end-users might discover on their own.

application_type: This is a 15-bit field which identifies the type of the applications described in this AIT sub-table. See table 66.

Table 66: Application types

application_type	description
0x0000	reserved_future_use
0x0001	DVB-J application
0x0002	DVB-HTML application
0x0003 to 0x7FFF	subject to registration with DVB

version_number: This 5-bit field is the version number of the sub-table. The version_number shall be incremented by 1 when a change in the information carried within the sub-table occurs. When it reaches value "31", it wraps around to "0".

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

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

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

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

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

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 clause 10.5, "Application identification".

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 [4] after processing the entire section.

10.4.7 Use of private descriptors in the AIT

Private descriptors may be included in the AIT provided that they are in the scope of a DVB-SI EN 300 468 [4] private data specifier descriptor. The scope rules for the private data specifier descriptor are as follows:

- If this descriptor is located within any descriptor loop of the AIT, then any specifier identified within this descriptor loop applies to all following descriptors and user-defined values in the particular descriptor loop until the end of the descriptor loop, or until another occurrence of a private data specifier descriptor.
- The use of the descriptor in the common (first) descriptor loop does not apply to descriptors or user-defined values in the application (second) descriptor loop.

10.4.8 Text encoding in AIT

Unless otherwise specified, all fields interpreted as text strings in the AIT shall be encoded as UTF8 (see clause 7.1.5, "Monomedia format for text"), but shall not include the null character. See also clause 14.5, "Text encoding of application identifiers".

10.4.9 AIT file

10.4.9.1 Syntax

The interaction channel encoding of the AIT into the AIT file is as follows:

- A single file shall contain all of the data.
- The file shall contain a concatentation of Application Information Sections (specified in clause 10.4.6 of the present document).
- The possibly multiple sections shall be ordered as follows:
  - Ascending order of `application_type`.
  - Within a single value of `application_type` in ascending order of `section_number`.
- All sections shall have `current_next_indicator` set to '1'.

10.4.9.2 Syntactic restrictions

10.4.9.2.1 Transport protocols

The only allowed transport protocol type has id value 0x0003. See table 76, "Protocol_id".

10.4.9.3 Semantics

The AIT file shall be loaded once during service selection. There is no requirement to monitor or poll the AIT file for any subsequent changes except as part of a subsequent service selection operation referring to the same AIT file. Consequences of this are:

- Only the AUTOSTART and PRESENT application control codes (of those defined in MHP 1.0) are appropriate.
- No standard mechanism for dynamic lifecycle control is provided.

NOTE: If dynamic lifecycle control is required application private mechanisms could be used. For example, a TCP connection over the interaction channel to a controlling server.

- Changes made to the AIT file after service selection shall not be detected or reported.

10.4.9.4 MIME type

The MIME type for an AIT file shall be "application/dvb.ait". The file extension shall be ".ait".

10.5 Application identification

10.5.1 Encoding

Each application is associated with an application identifier. This is a 6 byte field with the following structure.

Table 67: Application identifier syntax

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

organization_id: An `organization_id`, as defined in GEM [1], clause 10.4.3 under `organization_id`.

NOTE: The.note discouraging the use of `organization_id` values between 0x80000000 and 0xffffffff does not apply to the present document.

application_id: An `application_id`, as defined in GEM [1], clause 10.4.3 under `application_id`.

The same `application_identifier()` shall appear only once within the set of AIT subtables of the same `application_type` inside a service.

10.5.2 Effects on life cycle

The main concepts here are:

- On service change, currently running, previously broadcast, applications whose `service_bound_flag` is set to "0" shall (subject to resource restrictions) continue running if their application identifier is listed in the Application Information Table of the newly selected service.

- On service change, currently running, previously broadcast, applications whose `service_bound_flag` is set to "0" shall (subject to resource restrictions) continue running if their application identifier is suitably listed in the External application authorization 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 in the same service context. So, if an application is already running in a service context, then another instance of the same application shall not be launched in that same service context. This affects the behaviour with respect to the application launching API and autostart applications after service selection.
- If the application signalling for an application has the `service_bound_flag` set to "1", then the application is killed upon service selection.

See also annex S, "Application Listing and Launching".

When an application continues running after service change, it shall run as signalled by its AIT entry in the new service and not the former service excluding effects of transport protocol descriptors.

10.5.3 Authentication of application identification

See GEM [1], clause 12.5.6, "subject".

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 clause 9.1, "Broadcast MHP applications".

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 authorization 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 recognize the application shall continue in its current state.

When a change in these control codes causes a state change of a running MHP application, an `AppStateChangeEvent` shall be generated to all DVB-J applications which have registered to receive such events for the application concerned.

10.6.2.1 DVB-J application control codes

GEM [1], clause 10.4.3.1 is included in the present document.

10.6.2.2 DVB-HTML application control codes

The application control codes for DVB-HTML applications are listed in table 68.

Table 68: DVB-HTML application control code values

Code	Identifier	Semantics
0x00		Reserved_future_use.
0x01	AUTOSTART	The Application Entry Point of the DVB-HTML application is loaded. This is loaded into the user agent, and the DVB-HTML actor is created (in the Loading state) and the DVB-HTML application is started. When these steps are complete the DVB-HTML actor is in the Active state.
0x02	PRESENT	Indicates that the DVB-HTML application is present in the service, but is not autostarted.
0x03	DESTROY	When the control code changes from AUTOSTART or PRESENT to DESTROY, the DVB-HTML actor goes to the Destroyed state.
0x04	KILL	When the control code changes from AUTOSTART or PRESENT or DESTROY to KILL the DVB-HTML actor goes to the Killed state.
0x05	PREFETCH	As for AUTOSTART except that the DVB-HTML actor holds on entry to the Active state and waits for a trigger before completely transitioning to the Active state.
0x06	REMOTE	This identifies a remote application that is only launchable after service selection.
0x07 to 0xFF		Reserved_future_use.

See clause 9.3.2, "DVB-HTML Application Lifecycle".

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 clause 10.4.3, "Optimized AIT signalling.

When the MHP detects a change of the content of the application signalling descriptor, it shall acquire the new version of the AIT and respond accordingly.

The presence of the `application_type` and AIT_version subfields is optional. If not present then the AIT transmission and monitoring applies, see clause 10.4.2, "AIT transmission and monitoring".

Table 69: 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++){		
reserved_future_use	1	
application_type	15	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 15 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.

A single elementary stream may have more than one data broadcast id descriptor to indicate conformance with more than one data broadcast specification. In addition, more than one data broadcast id descriptor may be used to list additional application types within the scope of a particular 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 EN 300 468 [4] (illustrated in table 70). Where no "id specific data" is provided the descriptor just identifies the "principal" component of a data broadcast.

Table 70: 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 0x00F0 or 0x00F1, (see table 90, "Registry of constant values") the syntax of the data broadcast id descriptor is as shown in table 71. This extends the generic descriptor with an optional list of application types for which autostart applications may exist within the data broadcast. This list provides a hint to allow the MHP terminal to prioritize connection to a data broadcast when several are provided by the service. If no list is provided then the data broadcast id descriptor is silent on the types of autostart applications that may be carried by the data broadcast. If the application list is not empty, then the data broadcast shall not include autostart applications of application types other than those in the list. It is not required that the data broadcast always include autostart applications of all types in the list.

Table 71: 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++) {			
reserved_future_use	1		
application_type	15	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 TR 101 162 [i.2].

application_type: This 15 bit field indicates the type of the application (i.e. the engine or plug-in on which the application can be executed). See table 66.

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

(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 106, "Profile encoding" for the encoding of these values.

service_bound_flag: A service bound flag, as defined in GEM [1], clause 10.4.3.

visibility: A visibility field, as defined in GEM [1], clause 10.4.3.

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

application_priority: An application priority, as defined in GEM [1], clause 10.4.3.

transport_protocol_label: This 8-bit field identifies a transport protocol that delivers the application. See `transport_protocol_label` in clause 10.8.1, "Transport protocol descriptor".

If more than one protocol is signalled then each protocol is an alternative delivery mechanism. The ordering indicates the broadcaster's view of which transport connection will provide the best user experience (first is best). This may be used as a hint by MHP terminal implementations. It shall be evaluated only once during the life time of the application.

The protocol selection by the MHP terminal may depend on a variety of factors including user preferences and the performance of the transport connections to the terminal.

10.7.4 User information descriptors

The user information descriptors complement the "Application descriptor" by providing information suitable for presentation to the user (where the "Application descriptor" provides technical information for automatic use by the receiver).

These descriptors are defined for use in the application loop of the AIT.

10.7.4.1 Application name descriptor

Exactly one instance of this descriptor shall be included in the application information of an application. The application name shall distinguish the application and shall be informative to the user.

Table 73: 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 [13] three character language code of the language of the following bouquet name. Both ISO 639.2/B and ISO 639.2/T may be used.

Each character is coded into 8 bits according to ISO 8859-1 [14] and inserted in order into the 24-bit field.

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

application_name_char: This field carries a string (not null terminated) of characters encoded in accordance with annex A of EN 300 468 [4], with the modification that the first byte of a text field having the value 0x15 shall be interpreted as meaning UTF-8 encoding as defined by ISO/IEC 10646 [12].

10.7.4.2 Application icons descriptor

GEM [1], clause 10.4.3.2 is included in the present document.

10.7.5 External application authorization descriptor

The "common" (first) descriptor loop of the Application Information Table may contain zero or more `external_application_authorization_descriptors`. Each descriptor contains information about external applications that are allowed to continue to run with the applications listed in this Application Information Table sub-table but cannot be launched from this service. The external authorization applies to applications with the identified `application_identifier()` that are of the `application_type` identified by the AIT subtable where this descriptor is contained.

Table 74: External application authorization descriptor syntax

	No. of Bits	Identifier	Value
external_application_authorization_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 clause 10.5, "Application identification".

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

If the 0xffff or 0xfffe wildcard is used for the `application_id` within the `application_identifier()` and there are some applications from the same `organization_id` explicitly signalled in the application loop of the AIT, the priority for those applications shall be the one signalled in the `application_descriptor` (see clause 10.7.3).

See application_priority under clause 10.7.3, "Application descriptor".

10.7.6 Graphics constraints descriptor

GEM [1], clause 10.4.3.3 is included in the present document with the following modifications:

This descriptor may be present either in the inner (application) loop of an AIT in which case it applies to only that application or the outer (common loop) of an AIT in which case it applies to all applications signalled in that AIT sub-table.

NOTE: A consequence of the description of `can_run_without_visible_ui` is that the signalling of MHP 1.0 applications which intentionally do not have a visible UI needs to be modified if they are ever used in an context where SD graphics are not available. Such a context is not defined by the present document but could potentially exist in GEM terminal specifications or other MHP derivatives.

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 75: 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 registered here and in TR 101 162 [i.2].

Table 76: Protocol_id

Protocol_id	Description
0x0000	Reserved_future_use.
0x0001	MHP Object Carousel as defined in annex B "(normative): Object carousel".
0x0002	IP via DVB multiprotocol encapsulation as defined in EN 301 192 [5], TR 101 202 [i.6].
0x0003	Transport via HTTP over the interaction channel as described in clause 10.8.1.3, "Transport via interaction channel".
0x0004 to 0x00FF	Reserved for use by DVB.
0x0100 to 0xFFFF	Subject to registration in TR 101 162 [i.2].

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

selector_byte: Additional protocol specific information.

Table 77: Semantics of selector bytes

Protocol_id	Selector byte data
0x0000	Reserved_future_use.
0x0001	See clause 10.8.1.1, "Transport via OC".
0x0002	See clause 10.8.1.2, "Transport via IP".
0x0003 to 0xFFFF	Not defined in the present document.
0x0003	See clause 10.8.1.3, "Transport via interaction channel".
0x0004 to 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 78.

Table 78: 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 service that is different to the one carrying the AIT. Such applications shall not be autostarted by receivers but are visible (subject to the visibility field of the application descriptor) via an application listing API for possible launching by service selection (but not via an application launching API). When this bit is set, the following 3 fields (original_network_id, transport_stream_id and service_id) are included in the selector bytes. This flag shall be set to "0" when the transport connection is provided by the current service.

Applications with this flag set shall either have their application control code set to REMOTE (see GEM [1], table 15 and table 68 in the present document), or they shall have an application storage descriptor with "launchable_completely_from_cache" set to "1". (In the latter case they should be signalled as requiring a receiver that supports the MHP 1.1 profiles, so that they will be ignored by MHP 1.0 receivers).

Applications where remote_connection is "1" that also have an application storage descriptor with "launchable_completely_from_cache" set to "1" are a special case. If such an application is cached on the terminal, it can be launched in the usual way. There are no special restrictions on the control code for an application signalled in

this way - e.g. it could be PRESENT or even AUTOSTART. If the application is not cached on the terminal, it cannot be launched and the signalled control code will be ignored - it will always be treated as if it was REMOTE.

Remote applications can be cached and stored in the usual way if an application first tunes the network interface to the appropriate transport stream.

See clause 11.7.2, "Application discovery and launching APIs".

original_network_id: This 16 bit field identifies the DVB-SI original network id of the transport stream that provides the transport connection.

transport_stream_id: This 16 bit field identifies the MPEG transport stream id of the transport stream that provides the transport connection.

service_id: This 16 bit field identifies the DVB-SI service id of the service that provides the transport connection.

10.8.1.2 Transport via IP

When the protocol ID is 0x0002 the selector bytes in the Transport protocol descriptor shall be as shown in table 79.

This structure includes two important components of the data_broadcast_descriptor defined in EN 301 192 [5]. It provides all the information necessary for the MHP to acquire applications and application data components delivered by IP protocols. The profiles where this is an optional or mandatory feature are listed in clause 15, "Detailed platform profile definitions".

Table 79: 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
for(i=0; i<N; i++){		
URL_length	8	uimsbf
for(j=0; j<URL_length; j++){		
URL_byte	8	uimsbf
}		
}		

remote_connection: This and the associated 3 fields (original_network_id, transport_stream_id and service_id) have identical syntax and semantics to the fields with the same names under clause 10.8.1.1, "Transport via OC".

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 [18].

For URL using the "server" field including the host:port notation as defined in RFC 2396 [18], 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 [19].

NOTE: The present document intentionally does not define or require any URL format to be supported in this descriptor. Hence it cannot be used in an inter-operable way.

10.8.1.3 Transport via interaction channel

When the protocol ID is 0x0003 the selector bytes in the Transport protocol descriptor shall be as shown in table 80, "Syntax of selector bytes for interaction transport". This allows encoding of a number of URLs. For efficiency when encoding possibly many similar URLs the encoding divides the URL into a shared base part and a set of URL extensions. The set of URLs can identify ZIP (ZIP [30]) files, or base URLs ending in the "/" character, that encapsulate portions of the file system. See GEM [1], clause 6.4.1.1 "File system logical structure" for a description of this file system.

Multiple transport protocol descriptors with the protocol ID value 0x0003 and the same transport protocol label may be provided to define a larger set of URLs to describe the file system.

Table 80: Syntax of selector bytes for interaction transport

Syntax	Bits	Mnemonic
for(i=0; i<N; i++){		
URL_base_length	8	uimsbf
for(j=0; j<N; j++){		
URL_base_byte	8	uimsbf
}		
URL_extension_count	8	uimsbf
for(j=0; j<URL_extension_count; j++){		
URL_extension_length	8	uimsbf
for(k=0; k<URL_length; k++){		
URL_extension_byte	8	uimsbf
}		
}		
}		

URL_base_length: This 8-bit field provides the number of bytes in the base part of the URL.

URL_base_byte: These bytes form the first part of a HTTP URL conforming to HTTP 1.0 (see RFC 1945 [31]), or the first part of an HTTPS URL conforming to HTTPS (see RFC 2818 [45]) or the first part of another URL conforming to RFC 2396 [18]. Where an HTTP URL is found, http shall be used as in clause 6.3.7.2, "MHP profile of HTTP 1.0". Where an HTTPS URL is found, https shall be used as in clause 6.3.7.3, "HTTPS". For any other URL, the registered interaction channel transport protocol service provider implementation for the associated URL scheme will be used, if one exists. In the case where no provider exists for the scheme, the entire transport protocol descriptor will be ignored.

URL_extension_count: This 8-bit field indicates the number of URLs conveyed by this descriptor.

URL_extension_length: This 8-bit field indicates the number of bytes in the extension part of the URL.

URL_extension_byte: These bytes form the later part of an HTTP URL conforming to HTTP 1.0 (see RFC 1945 [31]), or the later part of an HTTPS URL conforming to HTTPS clause 6.3.7.3, "HTTPS" or else a URL whose scheme is supported by a registered interaction channel transport service provider implementation.

URLs are formed by concantenating the URL extension with the preceding URL base. The URL so formed either identifies a file system directory or a specific ZIP file. See GEM [1], clause 6.4.1.1 "File system logical structure".

10.8.2 IP signalling descriptor

The IP signalling descriptor is defined for use either in the "common" or in the "application" loop of the AIT. This descriptor indicates the identification of the organization providing the IP multicast streams used by all applications (when present in the "common" loop) or by the particular signalled application (when present in the "application" loop). See EN 301 192 [5] for the definition of the INT.

This descriptor and the INT with action_type 0x01 shall be used for applications relying on the presence of IP Multicast streams on the broadcast link. The knowledge of the identification present in the descriptor enables to recover the appropriate IP Notification Table (INT) with action_type 0x01 that contains the correspondence between the multicast IP address and port and the stream localization.

Table 81: Syntax of the IP signalling descriptor

Syntax	Bits	Mnemonic
ip_signalling_descriptor () {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
platform_id	24	uimsbf
}		

descriptor_tag: This 8-bit field with value 0x11 identifies this descriptor.

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

platform_id: This is a 24 bit field containing a `platform_id` of the organization providing IP/MAC streams on DVB transport streams/services.

Allocations of the value of `platform_id` are found in the TR 101 162 [i.2].

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 clause do not have a relation to the API-based pre-fetching for this version of the present document.

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

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

10.8.3.2 Pre-fetch descriptor

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

MHP terminals may use this descriptor to improve application start-up time by pre-fetching modules that have the indicated labels (see clause B.2.2.4.1, "Label descriptor").

Table 82: 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_byte	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 object carousel that delivers the modules to which this prefetch descriptor refers. See `transport_protocol_label` in clause 10.8.1, "Transport protocol descriptor".

label_length: This 8-bit field identifies the number of bytes in the module label.

label_byte: 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"). The match shall be done as a byte-by-byte comparison between the two byte arrays.

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 clause 10.8.3.2, "Pre-fetch descriptor"), 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 83: 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 object carousel that delivers the modules to which this prefetch descriptor refers. See `transport_protocol_label` in clause 10.8.1, "Transport protocol descriptor".

DII_identification: This 15-bit field identifies the DII message. It corresponds to the identification portion of the transactionId. See table B.33, "Sub-fields of the transactionId".

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 84: 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 a string that is passed to the application as a parameter.

NOTE: This somewhat exceeds the requirement of GEM [1], clause 10.5.2.

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 85: DVB-J application location descriptor syntax

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

descriptor_tag: This 8 bit integer with value 0x04 identifies this descriptor.

base_directory_length: This 8 bit integer specifies the number of bytes in the base_directory_byte string.

The value of this field shall be at least one.

base_directory_byte: These bytes contain a string specifying a directory name with directories delimited by the slash character "/" (0x2F). The directory name is relative to the top-level directory of the file system carrying the application as defined by the transport_protocol_descriptor for this application. This directory is used as a base directory for

relative path names. This base directory is automatically considered to form the first directory in the class path (after the path to the system's classes).

If the base directory is the root the string shall be "/".

classpath_extension_length: This 8 bit integer specifies the number of bytes in the classpath_extension_byte string.

classpath_extension_byte: These bytes contain a string specifying a further extension for the DVB-J class path where the classes of the application are searched in addition to the base directory.The class path extension string contains path names where the elements in the path are delimited by the semicolon character ";" (0x3B). The elements of the path may be either absolute paths starting from the root of the file system or they can be relative to the base directory. The directories are delimited by the slash character "/" (0x2F) and absolute path names begin with the slash character "/" (0x2F).

initial_class_byte: These bytes contain a string specifying the class implementing one of the Xlet interfaces specified in clause 11.7.1, "APIs to support DVB-J application lifecycle".

This string is a DVB-J class name that is found in the class path (e.g. "com.broadcaster.appA.MainClass"). The length of this string must be at least one.

NOTE: Since the classpath may only contain directories, the initial class is present in the form of a Java class file. A jar file cannot be referenced from the DVB-J application location descriptor.

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 86: DVB-HTML application descriptor syntax

	No. of Bits	Identifier	Value
dvb_html_application_descriptor(){			
descriptor_tag	8	uimsbf	
descriptor_length	8	uimsbf	
appid_set_length	8	uimsbf	N1
for(i=0; i<N1; i++) {			
application_id	16	bslbf	
}			
for(j=0; j<N; j++) {			
parameter_bytes	8	uimsbf	
}			
}			

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

appid_set_length: This 8 bit integer specifies the length of the list of `application_id`s.

application_id: The values of these 16 bit fields form a set of application ids (see clause 10.5, "Application identification"). This set is the set of application ids that are allowed to be associated with inner applications of a DVB-HTML application, see clause 8.9, "Xlet integration".

parameter_bytes: The parameter bytes contain the string that is appended to the application initial path as parameters.

The parameter bytes are joined using simple concatenation, it is the authors responsibility to ensure it is prefixed by a legal joining character (such as ? or #) to form a syntactically correct URL.

10.10.2 DVB-HTML application location descriptor

One instance of this descriptor shall be contained in the "application" (inner) descriptor loop of the AIT for each DVB-HTML application.

Table 87: 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	8	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 88.

Table 88: Transport specific semantic of physical root bytes

Protocol_id	Semantic
0x0000	Reserved_future_use.
0x0001	A directory specification.
0x0002	One of the base URLs defined in the Transport protocol descriptor signalled for the application (see clause 10.8.1.2, "Transport via IP").
0x0003 to 0xFFFF	TBD.

initial_path_bytes: These bytes contain a string specifying the URL path component to the entry point document. This path is relative to the root defined in the physical_root_bytes field.

10.10.2.1 Example

The following example describes the usage of the DVB-HTML application location descriptor.

An application author designs an HTML application in the following manner:

- The application data is distributed among several directories, let say an "`image`" directory and a "`main`" directory.
- The application entry point is an HTML document called "`index.htm`" and stored in the "`main`" directory.

10.10.2.2 Application Entry Point

From the application author's point of view, the application entry point is specified by the path "`main/index.htm`". This path is stored in the initial_path_bytes string of the location descriptor.

If the broadcaster inserts this application in a file system sub-directory called "`application`", the physical_root_bytes content of the location descriptor will be the string "`application/`".

If the broadcaster uses a transport via IP for this application, they shall signal the used protocol and IP address in the Transport protocol descriptor associated with this application and the physical_root_bytes field shall contain the corresponding URL string.

10.10.3 DVB-HTML application boundary descriptor

This descriptor is defined for use in the application loop of the AIT. It provides a regular expression that describes the data elements that form the application.

This descriptor is optional. When absent, the application boundary defaults to the complete set of all content coming from the transport signalled in the Transport protocol descriptor associated with the application.

Multiple boundary descriptors can be used for the same application. In this case, the equivalent global regular expression is the OR combination (union) of the individual regular expressions.

Table 89: 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	
}			
for(i=0; i<N; i++) {			
regular_expression_bytes	8	uimsbf	
}			
}			

descriptor_tag: This 8 bit integer with value 0x0A identifies this descriptor.

label_length: This 8 bit integer specifies the length of the label_bytes string.

label_bytes: These bytes contain a string specifying the label that is associated with the set of data identified by the regular expression. This label can be used for pre-fetching in a transport specific manner.

regular_expression_bytes: These bytes contain a string specifying the regular expression that can generate all URLs that are in the domain of the application.

See clause 9.3.1.4.1, "Regular Expression Syntax".

10.11 Constant values

Table 90: Registry of constant values

Where used	Type	Value	Where Defined	Scope
Private data specifier descriptor	descriptor tag	0x5F	PSI and SI tables	SI
Data broadcast id descriptor		0x66	PMT	
Application Signalling Descriptor		0x6F	PMT	
Service identifier descriptor		0x71	SDT	
Label descriptor		0x70	DII moduleInfo	SI-DAT
Caching priority descriptor		0x71	userInfo	
Content type descriptor		0x72	BIOP objectInfo (see note 1)	
Reserved to MHP for future OC descriptors		0x73 to 0x7F	OC	
Reserved to MHP for future use	table ID on AIT PID	0x00 to 0x73		MHP
Application Information Table		0x74		
Reserved to MHP for future use		0x75 to 0x7F		
Reserved for private use		0x80 to 0xFF		
Application descriptor	descriptor tag	0x00	AIT	MHP
Application name descriptor		0x01		
Transport protocol descriptor		0x02		
DVB-J application descriptor		0x03		
DVB-J application location descriptor		0x04		
External application authorization descriptor		0x05		
Reserved to MHP for future use		0x06 and 0x07		
DVB-HTML application descriptor		0x08		
DVB-HTML application location descriptor		0x09		
DVB-HTML application boundary descriptor		0x0A		
Application icons descriptor		0x0B		
Pre-fetch descriptor		0x0C		
DII location descriptor		0x0D		
Reserved to MHP for future use		0x0E to 0x10		
IP signalling descriptor		0x11		
Provider export descriptor (see table 93, "Provider export descriptor syntax")		0x12		
Provider usage descriptor (see table 94, "Provider usage descriptor syntax")		0x13		
Graphics constraints descriptor (see clause 10.7.6, "Graphics constraints descriptor")		0x14		
Reserved to MHP for future use		0x15 to 0x5E		
Reserved to MHP for future use		0x12 to 0x5E		
Delegated application descriptor		0x0E		
Plug-in descriptor		0x0F		
Application storage descriptor		0x10		
Reserved to MHP for future use		0x12 to 0x5E		
Private data specifier descriptor (see note 2)		0x5F		
Subject to registration in TR 101 162 [i.2]		0x60 to 0x7F		
User defined (see note 3)		0x80 to 0xFE		
MHP Object Carousel	data broadcast id	0x00F0	PMT, AIT	SI
Reserved for MHP Multi Protocol Encapsulation		0x00F1		
MHP application presence		0x00F2	EIT, SDT	SI
Reserved to MHP for future use		0x00F3 to 0x00FE	PMT, AIT	SI
MHP Application Service (see clause 10.11.1)	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 clause 10.4.7, "Use of private descriptors in the AIT").

10.11.1 MHP Application Service

This service type should be used for services which contain at least one auto-start MHP application where this application is the main component of the service. If a service signalled with this type contains any broadcast audio or video, the navigator shall not start presenting them but leave this to the auto-start application.

10.12 Service Information

10.12.1 Service identifier descriptor

Zero or more service identifier descriptors may be included in the SDT description of a service. Each such descriptor defines a single textual identifier for the service. The syntax of this identifier is specified in clause 14.10.1, "Syntax of the textual service identifier".

A single service identifier can be assigned to services in different physical networks even if they have different `original_network_id` and `service_id`. A given service identifier shall only be associated with services that are considered to be the same service.

NOTE: It is up to the service provider to decide which services are "same" and which are not. For example, two services in two different networks where the service have the same programme content but different regional adverts could be generally considered to be the "same" service. However, this decision is entirely up to the service provider.

More than one service identifier may be allocated to a service instance.

Table 91: Service identifier descriptor

	No. of Bits	Identifier	Value
service_identifier_descriptor () {			
descriptor_tag	8	uimsbf	
descriptor_length	8	uimsbf	
for (i = 0; i < descriptor_length; i++) {			
textual_service_identifier_bytes	8	uimsbf	
}			
}			

descriptor_tag: This 8 bit integer with value 0x71 identifies this descriptor.

textual_service_identifier_bytes: These bytes contain the unique identifier for a service encoded using the normal encoding for text strings in DVB SI.

10.12.2 Data broadcast descriptor for MHP application announcement

The generic data broadcast descriptor is defined in EN 300 468 [4]. This clause defines the syntax and semantics of the selector bytes when the data broadcast id has the value 0x00F2 (see table 92). In this case the selector bytes provide a list of MHP applications and information about each application. Zero or more instances of this descriptor may be listed in the SDT or the EIT to identify MHP applications associated with the service or the event where the descriptor is present. This descriptor only indicates the association between the service or event and the applications. The location of each listed application shall be resolved through the AIT. This descriptor shall not list applications where the test_application_flag is (or will be) set in the corresponding entry in the AIT.

Table 92: Syntax of extended data broadcast descriptor - broadcast id 0xF2

	No. of Bits	Identifier	Value
data_broadcast_descriptor(){			
descriptor_tag	8	uimsbf	
descriptor_length	8	uimsbf	
data_broadcast_id	16	uimsbf	
component_tag	8	uimsbf	
selector_length	8	uimsbf	
for(i=0; i<selector_length; i++){			
organization_id	32	uimsbf	
application_id	16	uimsbf	
reserved_future_use	1	bslbf	
application_type	15	uimsbf	
application_profile_length	8	uimsbf	
for (j=0; j<N; j++){			
application_profile	16	uimsbf	
version.major	8	uimsbf	
version.minor	8	uimsbf	
version.micro	8	uimsbf	
}			
application_names_length	8	uimsbf	
for(j=0; j<N2;j++){			
ISO_639_language_code	24	bslbf	
application_name_length	8	uimsbf	
for(l=0; l<N3; l++){			
application_name_char	8	bslbf	
}			
}			
reserved_length	8	uimsbf	
for(j=0; i<N4; i++){			
reserved_future_use	8	bslbf	
}			
private_data_length	8	uimsbf	
for(j=0; j<N5; j++){			
private_data_byte	8	bslbf	
}			
}			
ISO_639_language_code	24	bslbf	
text_length	8	uimsbf	
for (i=0; i<text_length; i++){			
text_char	8	uimsbf	
}			
}			

10.12.2.1 Semantics of the data broadcast descriptor

The semantics for the following elements of the syntax are defined in EN 300 468 [4]:

descriptor_tag: For this 8-bit field see EN 300 468 [4].

descriptor_length: For this 8-bit field see EN 300 468 [4].

data_broadcast_id: For this 16-bit field see EN 300 468 [4]. This field has the value 0x00F2 (see table 90) when announcing MHP applications (regardless of the transport method(s) used for the MHP application and data).

component_tag: For this 8-bit field see EN 300 468 [4].

selector_length: For this 8-bit field see EN 300 468 [4].

The semantics for the following elements of the syntax are defined in the present document:

organization_id: This is 32-bit field encodes the `organization_id` of the application. See clause 10.5.1.

application_id: This is 16-bit field encodes the `application_type` of the application. See clause 10.5.1.

application_type: This is 15-bit field encodes the `application_type` of the application. See clause 10.4.6.

application_profile_length: This 8-bit field indicates the length of the application profile loop in bytes.

application_profile: This is 16-bit field encodes the application_profile of the application. See clause 10.7.3.

version.major: This is 8-bit field encodes the version.major of the application. See clause 10.7.3.

version.minor: This is 8-bit field encodes the version.minor of the application. See clause 10.7.3.

version.micro: This is 8-bit field encodes the version.micro of the application. See clause 10.7.3.

application_names_length: This 8-bit unsigned integer specifies the number of bytes in the following multilingual application names.

ISO_639_language_code: This is 24-bit field encodes the ISO_639_language_code of the application name. See clause 10.7.4.

application_name_length: This is 8-bit field encodes the application_name_length of the application name. See clause 10.7.4.

application_name_char: See application_name_char in clause 10.7.4.

reserved_length: This 8-bit unsigned integer specifies the number of reserved bytes that follow.

reserved_future_use: This is an 8-bit field.

private_data_length: This 8-bit unsigned integer specifies the number of private data bytes that follow.

private_data_byte: This is an 8-bit field.

The semantics for the following elements of the syntax are defined in EN 300 468 [4]:

ISO_639_language_code: For this 24-bit field see EN 300 468 [4].

text_length: For this 8-bit field see EN 300 468 [4].

text_char: For this 8-bit field see EN 300 468 [4].

10.13 Plug-in signalling

GEM [1], clause 10.7 is included in the present document with the following notes and modifications.

Two signalling scenarios are defined for delegated applications and plug-ins:

- Native signalling scenario.
- MHP signalling scenario.

Functional equivalents for plug-in signalling are not permitted by the present document.

10.13.1 Native signalling scenario

In this scenario it is sufficient for the plug-in to be signalled as a normal MHP application. Optionally it could also be signalled as a plug-in using the Plug-in descriptor (see clause 10.13.4).

10.13.2 MHP signalling scenario

In this scenario MHP plug-in signalling can replace some or all of the native signalling of the delegated application. The MHP signalling allows the MHP terminal to:

- Identify that one (or more) delegated applications are available.
- The plug-in that is required.

- How to start the plug-in.
- How to introduce the delegated application to the plug-in.

The plug-in may use native signalling defined for the delegated application to locate, load and execute it. This is outside the scope of the present document.

Each delegated application is associated with an application descriptor (see clause 10.7.3, "Application descriptor"). The semantics of this descriptor are maintained with the following comments:

- The profile and version information is specific to the application format and are not registered by the MHP. However, the MHP rules for comparing profile and version values are maintained. So, the values in these fields are an encoding of the application's own profile and version numbering scheme to ensure that they are compatible with the MHP rules.
- The transport protocol label field is allowed to be empty if the delegated application is not transported by an MHP supported transport protocol.

The additional MHP signalling provided to support plug-ins is:

- "Delegated application descriptor" (see clause 10.13.3).

  An optional descriptor for delegated applications.

- "Plug-in descriptor" (see clause 10.13.4).

  A mandatory descriptor for plug-ins using MHP signalling.

10.13.3 Delegated application descriptor

GEM [1], clause 10.7.3 is included in the present document with the following notes and modifications.

Zero or one instance of this descriptor can be placed in either the common or application loops of the AIT of the delegated application.

10.13.4 Plug-in descriptor

GEM [1], clause 10.7.4 is included in the present document with the following notes and modifications:

One or more plug-in application descriptors shall be placed in the application loop of each plug-in.

The plug-in shall be considered suitable for running content if the `application_type` matches, and the version numbers are compatible according to the algorithm specified in clause 10.7.3, "Application descriptor" of the present document.

10.14 Stored applications

GEM [1], clause 10.8 is included in the present document with the following notes and modifications.

10.14.1 Use of signalling defined in MHP 1.0

Storable applications follow the standard MHP 1.0 application signalling. An MHP terminal that does not provide storage or does not recognize these extensions will perceive storable applications as viable applications that potentially can be run from the broadcast connection.

So, the signalling described here augments the signalling described in MHP 1.0.

Functional equivalents for plug-in signalling are not permitted by the present document.

10.14.1.1 Stored broadcast service related applications

For stored broadcast service related applications the broadcast AIT information shall be used when launching the stored application. So, little information from the AIT needs to be stored with the application.

10.14.1.2 Stored stand-alone applications

For stored stand-alone applications, the AIT information used when launching the application shall come from a stored representation of the AIT. While a stored service is being presented in a service context, the following apply:

- The applications database shall be populated with information from this stored representation of the AIT for the applications forming part of that stored service.
- The implementation shall report any changes in the stored representation of the AIT in the same way as it would report changes in a broadcast AIT.

10.14.2 Application storage descriptor

GEM [1], clause 10.8.2 is included in the present document with the following notes and modifications.

References in this clause to the Application Description File shall be interpreted as referring to clause 10.14.3 of the present document.

For a storable application a single application storage descriptor shall be placed in either the common descriptor loop or application descriptor loop of the AIT.

storage_property: Applications with this property may also be launched as if broadcast related if the currently selected service lists them in its AIT.

launchable_completely_from_cache: This field indicates that this application can be run entirely from cache, without connecting to the transport protocol signalled in the application's AIT, if all the critical files have been cached.

priority: The "Description" field of GEM [1], table 29, "Storage descriptor flag combinations", shall be modified as follows: the "normal case" storage descriptor flag combination (0, 0, 0) represents the MHP 1.0 equivalent.

10.14.3 Application Description File

10.14.3.1 Description

GEM [1], clause 10.8.3.1 is included in the present document.

10.14.3.2 Application Description File name and location

GEM [1], clause 10.8.3.2 is included in the present document with the following notes and modifications.

The application description file shall be located in the base directory of a DVB-J application (as signalled in the dvb-j application location descriptor) or the physical root of a DVB-HTML application (as signalled in the DVB-HTML application location descriptor).

10.14.3.3 Syntax

GEM [1], clause 10.8.3.3 is included in the present document.

10.14.3.4 Semantics

GEM [1], clause 10.8.3.4 is included in the present document with the following notes and modifications.

version: A decimal integer denoting the version number of this application.

Must not contain leading zeros (unless it is "0"). Must match the version number signalled in the version field of the application storage descriptor in the AIT entry of this application otherwise the application description file is invalid. (This field allows application authors to ensure that the version number signalled in the AIT is correct. If it is wrong then this prevents any files being stored.)

10.15 Signalling for providers

The provider export descriptor is used in the application (inner) loop of the AIT for an application which exports (contains) an XletBoundProvider. It declares that the classes of the XletBoundProvider are found in that application. This descriptor is meaningless if used in the common (outer) loop of the AIT.

Table 93: Provider export descriptor syntax

	No. of Bits	Identifier	Value
provider_export_descriptor() {			
descriptor_tag	8	uimsbf	0x12
descriptor_length	8	uimsbf	
For(i=0;i<N;i++) {			
provider_name_length	8	uimsbf	
For(j=0; i<N; i++) {			
provider_name_byte	8	uimsbf	
}			
provider_class_name_length	8	uimsbf	
For(j=0; i<N; i++) {			
provider_class_name_byte	8	uimsbf	
}			
}			
}			

Where the fields have the following meanings:

provider_name_length: This 8 bit unsigned integer specifies the number of bytes in the provider name.

provider_name_byte: These bytes contain a string specifying the name of the provider.

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

provider_class_name_byte: These bytes contain a string specifying the fully qualified name of the class in the application that implements the named provider.

The provider usage descriptor is used in either the application (inner) or common (outer) loop of the AIT for an application which uses an XletBoundProvider and does not include the classes of the provider as part of itself.

Table 94: Provider usage descriptor syntax

	No. of Bits	Identifier	Value
provider_usage_descriptor() {			
descriptor_tag	8	uimsbf	0x13
descriptor_length			
For(i=0;i<N;i++) {			
provider_name_length			
For(j=0; i<N; i++) {			
provider_name_byte			
}			
}			
}			

Where the fields have the following meanings:

provider_name_length: This 8 bit unsigned integer specifies the number of bytes in the provider name.

provider_name_byte: These bytes contain a string specifying the name of the provider.

10.16 Signalling for IPTV

10.16.1 Service bound application signalling

Service bound applications shall be signalled by including either an ApplicationList or an ExternalapplicationAuthorisedList in either the IPService or the Package elements of SD&S (see TS 102 034 [50]). This is fully specified in annex AR, "XML encoding for AIT" of the present document.

10.16.2 XAIT

Unbound applications shall be signalled by including an ApplicationList in the SD&S service provider discovery record for the subsidizing service provider. This is fully specified in annex AR, "XML encoding for AIT". Any ApplicationList elements in the service provider discovery records for other service providers shall be ignored.

Monitoring for changes in the XAIT shall be performed as defined in clause 5.4.3 of TS 102 034 [50].

NOTE: See org.ocap.application.AppSignalHander for notification of XAIT updates to applications.

10.17 XAIT for classical DVB networks

This XAIT signalling method is intended for operators and terminals that rely only on the DVB broadcast medium for signalling of the XAIT table. The terminal cannot rely on the TCP/IP return channel being available and therefore cannot use the XAIT XML signalling method as defined for IPTV in clause 10.16, "Signalling for IPTV".

Two variations on XAIT signalling are defined in this clause. Clauses 10.17.1 to 10.17.4 define the recommended approach. Clauses 10.17.5 and 10.17.6 define a still conformant but deprecated approach.

10.17.1 XAIT Definition

As in OCAP [49], the XAIT is signalled on a dedicated fixed PID. The default value of this PID is 0x1FFC, the same as defined in OCAP [49]. However this PID value can be defined by the operator through the `xait_pid_descriptor`. If an `xait_pid_descriptor` is signalled on the network by the operator, the terminal shall use the PID value for the XAIT that is defined in this `xait_pid_descriptor`. If there is no `xait_pid_descriptor` signalled on the network or when it is not found by the terminal at channel scanning time, the terminal shall use the default PID value of 0x1FFC.

In addition to interpreting the XAIT table, the terminal must also monitor XAIT version changes on the XAIT PID, even after transport stream changes, and act accordingly.

To ensure that the terminal can see all XAIT version changes immediately and independently of the currently tuned transport stream, the network operator should replicate the same XAIT table with the same PID value across all transport streams of its network.

If, after a service selection towards another transport stream, for some reason no XAIT table can be found on the previously defined XAIT PID value, the terminal shall assume that there was no version change to the last correctly signalled XAIT table and act accordingly.

NOTE 1: In the OCAP environment, it is not necessary to replicate the XAIT table across all transport streams because there the XAIT PID is always available on the dedicated out of band channel.

Clause 11.2.2.3, "OCAP XAIT" of OCAP [49] shall be supported except for the following clauses which are not required by the present document:

- 11.2.2.3.1 Application_Id Values.

NOTE 2: The present document uses a different range for privileged applications, see GEM [1], table 13, "Value ranges for application_id".

- 11.2.2.3.9 Transport via Interaction Channel.

- 11.2.2.3.12 OCAP-J Application Descriptor which shall be replaced by clause 10.9.1, "DVB-J application descriptor" of the present document.
- 11.2.2.3.13 OCAP-J Application Location Descriptor which shall be replaced by clause 10.9.2, "DVB-J application location descriptor" of the present document.

In addition to the AIT descriptors listed in clause 11.2.2.3 "OCAP 1.0 XAIT" of OCAP [49], any AIT descriptor used in the present document may be used in an XAIT. For storage of unbound applications the use of the OCAP-defined "Application Storage Descriptor" and "Application Description File" is recommended over the MHP-defined ones.

Signalling the location of the unbound applications located on an object carousel through the use of the `transport_protocol_descriptor` in the XAIT table and the expected behaviour is somewhat different than that of an OCAP XAIT and a regular AIT table. It is therefore further clarified in the next section.

10.17.2 Signalling of transport via object carousel

To signal the location of unbound applications in the XAIT, one must use the `transport_protocol_descriptor` as defined in 10.8.1, "Transport Protocol Descriptor", with the following characteristics:

- `protocol_id` = 0x0001 (which signifies "transport via OC").
- The "`remote_connection`" bit must be set to "1". Because the object carousel can be located on a transport stream other than the currently tuned transport stream, and because the XAIT table must be the same across all transport streams, only the unique triplet combination of `original_network_id`, `transport_stream_id` and `service_id` can be used to locate the object carousel consistently.

The triplet (`original_network_id`, `transport_stream_id`, `service_id`) uniquely defines the service containing the object carousel with the unbound applications. This service is typically a DVB data service.

The operator must ensure the following:

- This service contains the correct descriptors in the PMT as expected from any other broadcasted MHP DSMCC carousel (`carousel_identifier_descriptor`, `stream_identifier_descriptor`, `data_broadcast_id_descriptor`, `association_tag_descriptor`). Note that the PID of the XAIT should not be in the PMT of this service.
- The `component_tag` in the `transport_protocol_descriptor` of the XAIT matches the `component_tag` in the `stream_identifier_descriptor` signalled in the PMT elementary stream loop of the object carousel containing the unbound applications.

Upon loading of the unbound applications from the carousel, the terminal must tune to this triplet, mount the correct DSM-CC carousel (signalled through the corresponding `component_tag`), and fetch the unbound applications. The procedure and conditions for accomplishing this are described in "the application is located in a transport location in an inband channel carousel", defined in OCAP1.0.2 section 20.2.2.1.2, "Unbound Application Loading and Launching".

The applications in the XAIT are signalled with the "`remote_connection`" flag set to "1". In an exception to what is stated in clause 10.8.1.1, "Transport via OC" of the present document for applications with the `remote_connection` flag set to "1", these XAIT-signalled applications need not have an application control code set to `REMOTE`. There are no special restrictions on the control code for an application signalled in the XAIT - e.g. it could be `PRESENT` or even `AUTOSTART` and the terminal shall follow the application lifecycle behaviour as defined in OCAP [49].

10.17.3 xait_pid_descriptor

The `xait_pid_descriptor` is used to specify the PID value of the XAIT table for this network.

Table 95: xait_pid_descriptor

	No of Bits	Identifier	Value
xait_pid_descriptor() {			
descriptor_tag	8	uimsbf	**0x7f**
descriptor_tag_extension	8	uimsbf	**0x0c**
descriptor_length	8	uimsbf	
xait_PID	16	uimsbf	
}			

The maximum PID value is 0x1FFF (13 bit). The OCAP value for the XAIT PID is 0x1FFC.

The present document intentionally does not specify the location or locations in which this descriptor may be signalled. It may be dependent upon the scanning algorithms used, which are out of scope of the present document. For one particular scanning algorithm, a possible location for this descriptor would be the first descriptor loop of a `network_id` in the NIT table of all transport streams in the network.

If there is no `xait_pid_descriptor` signalled on the network, the terminal shall use the default PID value of 0x1FFC for the XAIT.

10.17.4 registerUnboundApp (xait)

When the "privileged application option" is supported by the terminal, loading/storing an unbound application through the monitor application using the method AppManagerProxy.registerUnboundApp("xait") as defined in OCAP [49], clause 10.2.2.3.2 "Applications registered by a privileged application" must be supported and must follow the behaviour specified by OCAP.

10.17.5 XAIT Definition (deprecated)

In classical DVB networks, the XAIT is in fact the AIT in a special service. Where the XAIT signals applications using a transport protocol descriptor with a `protocol_id` of 1, this shall refer to applications carried in carousels in that special service.

Clause 11.2.2.3 "OCAP XAIT" of OCAP [49] shall be supported except for the following sub-clauses which are not required by the present document:

- 11.2.2.3.1 Application_Id Values.

NOTE: The present document uses a different range for privileged applications, see GEM [1] table 13, "Value ranges for application_id".

- 11.2.2.3.9 Transport via Interaction Channel.
- 11.2.2.3.12 OCAP-J Application Descriptor which shall be replaced by clause 10.9.1, "DVB-J application descriptor" of the present document.
- 11.2.2.3.13 OCAP-J Application Location Descriptor which shall be replaced by clause 10.9.2, "DVB-J application location descriptor" of the present document.
- 11.2.2.3.17 Application Storage Descriptor.

In addition to the AIT descriptors listed in clause 11.2.2.3 "OCAP 1.0 XAIT" of OCAP [49], any AIT descriptor used in the present document may be used in an XAIT.

10.17.6 XAIT Discovery (deprecated)

The following descriptor is defined to specify the location of the service carrying the XAIT information.

Table 96: XAIT location descriptor syntax

	No. of Bits	Identifier	Value
xait_location_descriptor() {			
descriptor_tag	8	uimsbf	0x7d
descriptor_length	8	uimsbf	
xait_original_network_id	16	uimsbf	
xait_service_id	16	uimsbf	
xait_version_number	5	uimsbf	
xait_update_policy	3	uimsbf	
}			

Where the fields have the following meanings:

xait_original_network_id: The original network id of the service containing the XAIT.

xait_service_id: The service_id of the service containing the XAIT.

xait_version_number: The version number of the XAIT referenced by this descriptor.

xait_update_policy: The following table defines the values which this field can signal.

Table 97: Values for the xait_update_policy field

Value	Meaning
0	When the XAIT version changes, immediately re-load the XAIT
1	Ignore XAIT version changes until a reset or reinitialize
2 to 7	Reserved for future use

The present document intentionally does not specify the location or locations where this descriptor may be signalled. It may be very dependent on the scanning algorithms used which are out of scope of the present document. For one particular scanning algorithm, a possible location where this descriptor could be placed would be the NIT_other table of all transport streams in the network.

10.18 Device-resident unbound applications

A subsidizing service provide may require these to be present in the terminal by means outside the scope of the present document, e.g. at the time the device is manufactured or by software download as part of the system software of the terminal. No signalling is defined for such applications since the information that would be communicated by signalling is instead communicated through internal interfaces between two parts of the system software of the terminal.

Where these application are used, they shall comply with GEM [1], clause 11.2.3, "Class Loading" such that they run within their own classloader. Their classes shall not be visible to other MHP applications.

11 DVB-J Platform

11.1 The Java Platform

GEM [1], clause 11.1 is included in the present document.

11.2 General issues

GEM [1], clause 11.2 is included in the present document.

11.3 Fundamental DVB-J APIs

GEM [1], clause 11.3 is included in the present document.

11.4 Presentation APIs

11.4.1 Graphical User Interface API

GEM [1], clause 11.4.1 is included in the present document, with the following notes and modifications.

- GEM [1], clause 11.4.1.2 specifies that the packages `org.havi.ui` and `org.havi.ui.event` defined in HAVi [21] shall be supported with some exceptions. For MHP terminals, all HAVi UI widgets and supporting classes are required.

GEM [1], clause 11.4.1.5.2 is included in the present document, with the following notes and modifications.

- GEM [1], clause 11.4.1.5.2, "OpenType font bindings" is included with the addition of the following text:

  All font metrics values shall be converted from font metrics units to pixels as described in clause GEM [1], clause D.3.4 "Converting font metrics to display pixels" before being returned. This conversion shall round up.

  For OpenType fonts, the values `metricsResolution` and `outlineResolution` shall be equal to the value of `unitsPerEm` field, defined in the Font Header ('head') table.

  The "advance width" of a character in font metrics units is defined as follows:

  - For proportional fonts, where the value of `isFixedPitch` defined in the `PostScript ('post')` table of OpenType/TrueType font equals to '0' (i.e. `false`), the advance width for each glyph is given in the `Horizontal Metrics ('hmtx')` table array.
  - For monospaced fonts, with the value of `isFixedPitch` set (non-zero), the advance width of a character is determined by the single entry in the Horizontal Metrics ('hmtx') table.

  `java.awt.FontMetrics.charWidth()` shall return the advance width of the character converted to pixels. This conversion shall round up.

  `java.awt.FontMetrics.stringWidth()`, `java.awt.FontMetrics.bytesWidth()` and `java.awt.FontMetrics.charsWidth()` are calculated by summing the advance widths of all the characters in the string, and adjusting for any kerning in the font. Adjustments for kerning are performed in metrics units, and then the result is converted to pixel units. This conversion shall round up.

  `java.awt.FontMetrics.getMaxAdvance()` returns the maximum value that `java.awt.FontMetrics.charWidth()` can return. This value shall be obtained from the `advanceWidthMax` field of Horizontal Header ('hhea') table of OpenType font.

11.4.2 Streamed Media API

GEM [1], clause 11.4.2 is included in the present document, with the following notes and modifications.

The text in GEM [1], clause 11.4.2.3 item b defines a default for service component presentation, then allows for it to be overridden:

- The streams which are first in the network signalling information (i.e. in the PMT) will be presented. GEM terminal specifications may define additional mechanisms that override this default.
- The second sentence, "GEM terminal specifications may define additional mechanisms that override this default" does not apply to MHP terminals; that is, the behaviour described is the default.

The following classes exempted by GEM [1], clause 11.4.2.5.1 are required by the present document:

- `org.dvb.media.SubtitlingEventControl;`

- `org.dvb.media.SubtitleAvailableEvent;`
- `org.dvb.media.SubtitleListener;`
- `org.dvb.media.SubtitleNotAvailableEvent;`
- `org.dvb.media.SubtitleNotSelectedEvent;`
- `org.dvb.media.SubtitleSelectedEvent;`
- `org.dvb.media.CAStopEvent;`
- `org.dvb.media.CAException.`

The following class is not required by GEM [1], clause 11.4.2.5.2 but is required by the present document:

- `org.davic.media.SubtitlingLanguageControl.`

GEM [1], clause 11.4.2.9 defines a mapping between the provider APIs and the corresponding Java TV API.

The Provider API (clause 11.7.9 in the present document) is required in MHP.

The `DVBTextLayoutManager` specified in annex U, "Extended graphics APIs" shall be supported.

11.5 Data Access APIs

GEM [1], clause 11.5 is included in the present document, with the following notes and modifications.

11.5.1 Broadcast Transport Protocol Access API

GEM [1], clause 11.5.1 is included in the present document, with the following notes and modifications.

The reference to annex P is to be read as referring to annex P of the present document.

Relative file names used to access objects in the carousel shall be taken as being relative to the base directory indicated in clause 10.9.2, "DVB-J application location descriptor". Calling `new DSMCCObject(".")` or `new java.io.File(".")` will instantiate the directory object that refers to the base directory as indicated in clause 10.9.2, "DVB-J application location descriptor".

In GEM [1], clause 11.5.1.1, the fifth bullet point, which addresses the method `lastModified()` is replaced by:

- The method `lastModified()` returns moduleVersion from the DII for the module that carries the file (treating the octet as an unsigned integer).

11.6 Service Information and Selection APIs

11.6.1 DVB Service Information API

The DVB specific SI API is defined in annex M, "SI Access API".

11.6.2 Service Selection API

GEM [1], clause 11.6.2 is included in the present document.

11.6.3 Tuning API

GEM [1], clause 11.6.3 is included in the present document, with the following notes and modifications.

The reference to GEM [1], clause 11.7.6 is to be read as referring to clause 11.7.6 of the present document.

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 clause 11.2.2 - 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 shall throw `java.lang.SecurityException` if the calling application does not have a CAPermission as defined in the description of the CAPermission class and shall not throw that exception if the calling application does have such a CAPermission.

- `CAModuleManager.addMMIListener;`
- `CAModule.queryEntitlement;`
- `CAModule.listEntitlements;`
- `CAModule.buyEntitlement;`
- `CAModule.openMessageSession.`

The `sessionIDs` used in CI message passing are unique to a single application and access shall fail if shared between applications.

See also clause 11.8.3, "Additional permissions classes".

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

11.6.5 Protocol Independent SI API

GEM [1], clause 11.6.5 is included in the present document.

11.6.6 Service discovery and selection for IPTV

GEM [1], clause 11.6.6 is included in the present document.

11.6.7 Integration between protocol independent SI API and TV-Anytime

GEM [1], clause 11.6.7 is included in the present document.

11.7 Common Infrastructure APIs

11.7.1 APIs to support DVB-J application lifecycle

GEM [1], clause 11.7.1 is included in the present document, with the following notes and modifications.

In clause 11.7.1.1, MHP requires that a specific text encoding be used:

Each entry in the loop of that descriptor shall be presented as one string in the array returned by this method, interpreted using the encoding as specified in clause 10.4.8, "Text encoding in AIT".

11.7.2 Application discovery and launching APIs

This API is formed of the `org.dvb.application` package defined in annex S "(normative): Application Listing and Launching".

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

Table 98: Application attribute properties

Property name (note)	Return
dvb.j.location.base	Returns String containing base_directory_bytes from DVB-J application location descriptor.
dvb.j.location.cpath.extension	Returns String[] derived from classpath_extension_bytes of DVB-J application location descriptor with each array entry corresponding to a pathname entry as defined for classpath_extension_bytes.
dvb.transport.oc.component.tag	Returns Integer containing the component_tag from the selector bytes of the transport protocol descriptor. If more than one transport protocol descriptor is in the AIT for a remote application then it is implementation dependent which of them is returned.
dvb.transport.oc.component.tag	Returns Integer containing the component_tag from the selector bytes of a transport protocol descriptor with `protocol_id` 0x0001. If more than one transport protocol descriptor with `protocol_id` 0x0001 is in the AIT for a remote application, it is implementation dependent which of them is returned.
dvb.ait.descriptors	See clause 11.7.8, "Plug-in APIs".
dvb.transport.oc.locator	An org.davic.net.Locator that identifies the object carousel. This shall be a Locator that is suitable for the ServiceDomain.attach(Locator) method. This shall be the Locator for the component specified by the "dvb.transport.oc.component.tag" property, in the service identified by getServiceLocator().
NOTE: Property names beginning "dvb." are reserved for future use.	

Where a property is specified as either containing a value from a particular descriptor or being derived from a particular descriptor, the property shall not be returned for applications for which that descriptor is not signalled. Where a property is specified as containing a value from a transport protocol descriptor with a particular `protocol_id`, the property shall not be returned for applications for which a transport protocol descriptor with that `protocol_id` is not signalled.

NOTE: The three properties `dvb.j.location.base`, `dvb.j.location.cpath.extension` and `dvb.transport.oc.component.tag`, defined above, are specific to transport via an MHP object carousel. They may not be available if other transport is used in other specifications, e.g. in a GEM terminal specification (see GEM) or a future version of the present document.

The following table defines the source of the information which shall be used for methods returning information from entries in the application database for an application signalled in an AIT. This shall apply to both AITs delivered via MPEG-2 broadcast and ones delivered as an AIT file (see clause 10.4.9, "AIT file"). This shall also apply to applications running as part of a stored service using information stored at the time when the application itself was stored.

Table 99: Information source for methods on AppAttributes

Method	Information source
getName()	One of the names that can be found in the application name descriptor.
getName(String ISO639code)	A name of the application from the application name descriptor corresponding to the specified language.
getNames()	All of the names for the application which can be found in the application name descriptor and their ISO 639 language code.
getProfiles()	The set of profiles signalled in the application profile field of the application descriptor.
getPriority()	The contents of the application_priority field of the application descriptor.
getVersions(String profile)	The values version.major, version.minor and version.micro for the specified profile from the application descriptor.
getIsServiceBound()	True if the service_bound field in the application descriptor is set to 1. Otherwise false.
isStartable()	There is no information source for this method, the return value is derived as specified in the method description. For the purpose of the method description, remote applications are defined to be those signalled as such in the transport protocol descriptor.
getIdentifier()	The contents of the application_identifier field of the application information section.
getServiceLocator()	An application shall be considered to be transmitted on a remote connection only where the application control code in the signalling is REMOTE. The locator for a remote application shall encapsulate the values found in the selector bytes of the transport protocol descriptor.
getType()	Returns the `application_type` field from the AIT section in which the application was signalled.
isVisible()	True if the visibility field in the application descriptor is set to 11. False otherwise.

Table 100: Information source for methods on AppIcon

Method	Information source
getLocator()	The bytes carried in the application_locator_byte of the application icons descriptor appended to either the base directory of the application (where the application type from the application information section is zero, from the DVB-J application location descriptor) or the physical root (where the application type from the application information section is 1, from the DVB-HTML application location descriptor).
getIconFlags()	The icon_flags field of the application_icon_descriptor.

Applications signalled in an AIT shall be considered to be externally authorized where their only signalling in that AIT is by an External application authorization descriptor.

`CurrentServiceFilter` shall behave as defined in its specification for stored services and services signalled over the interaction channel in addition to broadcast service as defined in its specification.

11.7.3 Inter-Application and Inter-Xlet communication API

GEM [1], clause 11.7.3 is included in the present document.

11.7.4 Basic MPEG Concepts

GEM [1], clause 11.7.4 is included in the present document, with the following notes and modifications.

The following additional classes are required by the present document:

- `DvbElementaryStream;`
- `DvbService;`
- `DvbTransportStream.`

11.7.5 Resource Notification

GEM [1], clause 11.7.5 is included in the present document.

11.7.6 Content Referencing

This API is formed of the classes found in section H.4 of annex H of DAVIC 1.4.1p9 [3] - the `Locator` and `DvbLocator` classes. It also includes the `javax.tv.locator` package as defined in Java TV [51]. It also includes the `org.dvb.locator.package` defined in annex AL.

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

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

The `createFactory()` method of `javax.tv.locator.LocatorFactory` shall always return `org.davic.net.Locator(s)` which implement the `javax.tv.locator.Locator` interface when provided with DVB URLs as input (as defined in clause 14.1 "Namespace mapping (DVB Locator)").

In the present document, methods whose signature has a return type of `org.davic.net.Locator` or `javax.tv.locator.Locator` shall return an instance of `org.davic.net.dvb.DvbLocator` (or a platform defined subclass of that) where the locator returned can be represented by the DVB locator syntax described in DAVIC 1.4.1p9 [3]. In this case, the DvbLocator returned shall contain the numeric identifiers of a DVB service (see clause 14.10, "Service identification").

In `javax.tv.locator.Locator.toExternalForm()`, the canonical form of a DVB locator is defined as follows:

- For instances of `org.davic.net.dvb.DvbNetworkBoundLocator` or `org.dvb.locator.FrequencyLocator`, the syntax of this string is implementation dependent. However, the resulting string shall be sufficient to recreate an equivalent Locator (e.g. using Java TV's `LocatorFactory`) or `MediaLocator` at a later time. The same string shall be valid even if stored by the Xlet (e.g. in persistent storage) and used at a later time.

- For instances of `org.davic.net.dvb.DvbLocator` which are not instances of the above sub-class and reference a service or more detailed parts of a service, this shall be the format defined in the MHP specification. If the optional transport_stream_id was provided when the instance was built, it shall form part of the external form otherwise it shall not.

- For instances of `org.davic.net.dvb.DvbLocator` which are not instances of the above sub-class and reference only a transport stream but not a service, this shall be the format defined in the MHP specification, including the transport stream id.

NOTE: Because optional extensions (e.g. component tag, event id) are part of the external form, they are considered in a comparison thus if they are not equally present in both locators then the comparison will fail.

The protected constructor of `LocatorFactory` is for implementation use. MHP applications shall not subclass `LocatorFactory`. Implementations are not required to behave correctly if they should do this.

11.7.7 Common Error Reporting

GEM [1], clause 11.7.7 is included in the present document.

11.7.8 Plug-in APIs

GEM [1], clause 11.7.8 is included in the present document.

11.7.9 Provider API

GEM [1], clause 11.7.9 is included in the present document.

11.7.10 Content referencing for IPTV

GEM [1], clause 11.7.10 is included in the present document.

11.7.11 TV-Anytime content referencing and metadata

GEM [1], clause 11.7.11 is included in the present document.

11.8 Security

GEM [1], clause 11.8 is included in the present document, with the following notes and modifications.

11.8.1 Basic Security

GEM [1], clause 11.8.1 is included in the present document.

11.8.2 APIs for return channel security

GEM [1], clause 11.8.2 is included in the present document, with the following notes and modifications.

Cipher suites listed in GEM [1], table 56, "Profile of cipher suites that implementations are required to support" shall have the name found in the "cipher suite" column of that table with "TLS_" replaced by "SSL_". If other cipher suites from TLS RFC 2246 [24] are supported, it is implementation dependent whether their names start with "TLS_" or "SSL_".

NOTE: For MHP, this language might be interpreted as disallowing these cipher suites to also be reported with a name beginning with "`TLS_`".

11.8.3 Additional permissions classes

GEM [1], clause 11.8.3 is included in the present document.

11.8.4 General security issues

GEM [1], clause 11.8.4 is included in the present document.

11.8.5 Cryptographic API

GEM [1], clause 11.8.5 is included in the present document.

11.8.6 DVB Extensions for Cryptography

GEM [1], clause 11.8.6 is included in the present document.

11.9 Other APIs

GEM [1], clause 11.9 is included in the present document.

11.10 Java permissions

GEM [1], clause 11.10 is included in the present document, with the following notes and modifications.

In MHP, the only mechanism by which an application may be trusted, and thus request additional permissions, is for that application to be signed.

11.11 Content referencing

The following mapping shall be used between the types of locator defined in table 103, "Addressable entities, locators and their text representation" and the DVB-J methods defined in this clause. It lists the Java methods and constructors which accept or return (as defined by their method signature) instances of `org.davic.net.Locator`, `javax.tv.locator.Locator`, `javax.media.MediaLocator` or their sub-classes. The external form of these locators shall be the text representation defined in the table 103, "Addressable entities, locators and their text representation" for the corresponding entity being referenced. Where the same method is listed as accepting multiple forms of locator, then it is required to accept all forms listed in this clause.

Where a method listed below is defined (in its specification) to check its input then it shall only accept the forms of locator listed below as being valid for that method from among those defined in the present document. Other forms of locator from among those defined in the present document shall be rejected as specified for the method concerned. If a method does not specify a means of rejecting inappropriate locators then it shall fail silently apart from Exceptions and Events which do not check their input and where it is the responsibility of the platform to use correct locators when constructing them. The present document does not prevent methods accepting other forms of locator which are not defined in the present document.

The clauses below apply equally to locators including and not including the network_id, i.e. for methods accepting a locator including the network_id, the network id shall be matched against the corresponding field in the NIT.

11.11.1 Transport stream

GEM [1], clause 11.11.1 is included in the present document.

11.11.2 Network

GEM [1], clause 11.11.2 is included in the present document.

11.11.3 Bouquet

The following methods used in the present document shall accept or return instances of Objects which describe a DVB bouquet.

- `javax.tv.service.transport.BouquetCollection.retrieveBouquet();`
- `javax.tv.service.transport.Bouquet.getLocator().`

11.11.4 Service

11.11.4.1 MPEG/DVB specific service

GEM [1], clause 11.11.4.1 is included in the present document, with the following notes and modifications.

The term "GEM service" is to be interpreted as meaning "DVB service". "GEM locator" is to be interpreted as meaning "DVB locator".

The following methods *are* required by the present document:

- `org.davic.net.ca.CAModule.buyEntitlement();`
- `org.davic.net.ca.CAModule.queryEntitlement();`
- `org.dvb.si.SIDatabase.retrieveSIService();`
- `org.dvb.si.SIDatabase.retrievePMTService();`
- `org.dvb.si.PMTService.getDvbLocator();`
- `org.dvb.si.SIBouquet.getSIServiceLocators();`
- `org.dvb.si.SIService.getDvbLocator();`

- `org.davic.net.ca.TuneRequestEvent - constructor;`
- `org.davic.net.ca.TuneRequestEvent.getLocator().`

11.11.4.2 Generic service

GEM [1], clause 11.11.4.1 is included in the present document, with the following notes and modifications.

The term "GEM specific service" is to be interpreted as meaning "DVB service".

11.11.4.3 Content referencing for IPTV

GEM [1], clause 11.11.4.3 is included in the present document.

11.11.4.4 Stored services

GEM [1], clause 11.11.4.4 is included in the present document.

11.11.4.5 Internet client services

The following methods used in the present document shall accept or return instances of Objects which describe an internet client application.

- `org.dvb.internet.InternetClient.getServiceContentLocators;`
- `javax.tv.service.Service.getLocator` - when the Service is an `InternetClientService.`

11.11.5 DVB event

GEM [1], clause 11.11.5 is included in the present document, with the following notes and modifications.

The term "program Event" is to be interpreted as meaning "DVB event".

The following methods, which are optional in GEM, are required by the present document:

- `org.davic.net.ca.CAModule.buyEntitlement();`
- `org.davic.net.ca.CAModule.queryEntitlement();`
- `org.dvb.si.SIEvent.getDvbLocator().`

11.11.6 MPEG elementary stream

GEM [1], clause 11.11.6 is included in the present document, with the following notes and modifications.

The phrase "GEM locators including a reference to multiple components" is to be interpreted as meaning "DVB locators including multiple component tags." In the bulleted list, the note "shall also accept multiple component tag GEM locator" shall be interpreted as referring to these same DVB locators.

The following methods, which are optional in GEM, are required by the present document:

- `org.dvb.si.SIDatabase.retrievePMTElementaryStreams();`
- `org.dvb.si.PMTElementaryStream.getDvbLocator();`
- `org.davic.net.ca.DescramblingStoppedEvent.getServiceLocator();`
- `org.davic.net.ca.DescramblingStartedEvent.getServiceLocator().`

11.11.7 File

GEM [1], clause 11.11.7 is included in the present document, with the following notes and modifications.

The bullet item:

- `org.dvb.dsmcc.ServiceDomain.getURL(Locator)` - locators referring to File or Directory entities, as defined in table 103.

shall be interpreted as follows:

- `org.dvb.dsmcc.ServiceDomain.getURL(Locator)` - instances of "dvb:" locator including dvb_abs_path.

11.11.8 Directory

GEM [1], clause 11.11.8 is included in the present document, with the following notes and modifications.

The phrase "GEM locator" shall be interpreted as meaning "dvb:" locator.

11.11.9 Drip feed decoder

GEM [1], clause 11.11.9 is included in the present document.

11.11.10 Methods with no defined requirements

GEM [1], clause 11.11.10 is included in the present document.

11.11.11 Methods working on many Locator types

The following methods used in the present document work on many locator types. The locator types which each method is required to support are listed for each of the methods concerned.

- `javax.tv.locator.LocatorFactory.transformLocator` - transforms a transport independent locator into a transport dependent one.

  Required to accept instances of `org.davic.net.dvb.DvbLocator`.

  Required to return instances of `org.davic.net.dvb.DvbNetworkBoundLocator`.

- `javax.tv.locator.LocatorFactory.createLocator` - creates a locator from a string.

  When passed the appropriate text representation, required to return locators for all entities defined to be addressable by Locators in table 103, "Addressable entities, locators and their text representation" except for "Drip feed decoder".

NOTE: Where no text representation is defined, a String in the appropriate format can of course be obtained by calling `Locator.toExternalForm` on a Locator for the appropriate entity.

- `javax.tv.service.SIManager.registerInterest` - accepts a locator referencing one or more SIElements as input.

- `javax.tv.service.SIManager.retrieveSIElement` - accepts a locator referencing one or more SIElements as input.

  Both these methods are required to accept locators referencing:-Bouquet, Network, Event, ElementaryStream, Service, TransportStream.

- `javax.tv.service.SIElement.getLocator` - Returns a locator for "this SIElement" as specified by the Java TV specified sub-interfaces, no other SIElements exist.

11.11.12 Support for the HTTP protocol in DVB-J

GEM [1], clause 11.11.12 is included in the present document.

11.11.13 MHP applications

GEM [1], clause 11.11.13 is included in the present document.

11.12 Stand-alone applications

11.12.1 Common behaviour

GEM [1], clause 11.12.1 is included in the present document.

11.12.2 Stored services

GEM [1], clause 11.12.2 is included in the present document, with the following notes and modifications.

This feature is optional in GEM but is made mandatory in MHP.

11.13 Support for DVB-HTML

11.13.1 Document object model APIs

11.13.1.1 Framework

The required document object model APIs are defined in the following:

- From Document Object Model (DOM) Level 2 Core Specification [34], that part of annex D, "Java Language Binding" corresponding to the "Fundamental Interfaces" as specified in section 1.2 of that specification.
- From Document Object Model (DOM) Level 2 Events Specification [36], annex B, "Java Language Binding", only those parts corresponding to the interfaces listed as required in clause 8.11.1.1, "Fundamental interfaces" and clause 8.11.1.2, "Event interfaces".
- From Document Object Model (DOM) Level 2 Views Specification [38], annex B, "Java Language Binding".
- From Document Object Model (DOM) Level 2 Style Specification [37], annex B, "Java Language Binding" the interface org.w3c.dom.css.CSS2Properties is required.

11.13.1.2 DVB defined extensions

The DVB defined DOM extensions are defined in annex AD "(normative): Support for DVB-HTML" of the present document see also clause AD.1, "Java bindings to DVB extensions".

11.13.1.3 Read Only Access to DOM

Where a DVB-J application only has or requests read-only access to the DOM, calling methods in the DOM APIs whose result would be to modify the DOM shall result in an `org.w3c.dom.DOMException` with exception code `NO_MODIFICATION_ALLOWED_ERR` being thrown.

11.13.2 Embedded Xlets in a DVB-HTML Page

Below is a list of the semantic differences in MHP APIs between DVB-J applications and xlets embedded in a DVB-HTML page of a DVB-HTML application.

- Xlets shall not use the HAVi HScene APIs to get the "primary" Container that is managed by the enclosing DVB-HTML document. The method `javax.tv.graphics.TVContainer.getRootContainer(javax.tv.xlet.XletContext) or javax.microedition.xlet.XletContext.getContainer()` shall be used for this purpose. The object returned shall be a subclass of `java.awt.Container` implementing the following interfaces.

  - `org.havi.ui.HComponentOrdering.`
  - `org.dvb.application.inner.DVBScene.`

  It may be an instance of org.havi.ui.HScene or a subclass of that but is not required to be either of these.

- Embedded Xlets are only guaranteed to be able to perform actions using the Document Object Model APIs when they are in the active state. The results of calling methods on `org.dvb.dom.bootstrap.DocumentFactory` when the embedded xlet is in any other state are implementation dependent. They may include a RuntimeException being thrown or the Document passed to `DocumentAction.run()` being null. If the state of an application changes from the active state to another state between when a method on `DocumentFactory` is called and when the `run()` method of the specified `DocumentAction` is called then the results of this are also implementation dependent.

- If the Xlet is in the active state or if the `startXlet` method is executing, and the width and height are > 0, then `getRootContainer()` shall return a valid `Container` instance. If the width or height are £ 0, `getRootContainer` shall return `null`. In all other cases, situations and xlet states, the value returned is implementation dependent.

Any "param" elements of the <object> used to define an embedded xlet shall be accessible to the embedded xlet. The sequence of these arguments is mapped to `XletContext.ARGS` and shall be available to the Xlet, in the form of an array of Strings, by calling:

```
javax.tv.xlet.XletContext.getXletProperty(XletContext.ARGS)
```

11.14 Internet access

GEM [1], clause 11.14 is included in the present document.

11.15 APIs defined in OCAP

GEM [1], clause 11.15 is included in the present document with the following modification:

- This clause is required in MHP.

12 Security

12.1 Introduction

GEM [1], clause 12. is included in the present document, with the following comments and modifications.

12.1.1 Overview of the security framework for applications

GEM [1], clause 12.1.1 is included in the present document, with the following comments and modifications.

The last paragraph of this clause is not included in the present document, and is replaced by the following:

Applications that are signed shall be identified with an `application_id` from the signed applications range (see GEM [1], table 13, "Value ranges for application_id". Applications that are unsigned shall be identified with an application_id from the unsigned applications range. An application with an `application_id` from the signed applications range but that is not signed is considered to have failed authentication. An application with an `application_id` from the unsigned applications range is treated as unsigned even if the files might be transmitted with signatures.

12.1.2 Overview of return channel security

GEM [1], clause 12.1.2 is included in the present document.

12.1.3 MHP application signing framework

In MHP, the only mechanism by which an application may be trusted, and thus request additional permissions, is for that application to be signed.

12.2 Authentication of applications

GEM [1], clause 12.2 is included in the present document.

12.3 Message transport

GEM [1], clause 12.3 is included in the present document.

12.4 Detail of application authentication messages

GEM [1], clause 12.4 is included in the present document with the following modification:

- NOTE 2: Authorizing an application authentication mechanism other than MHP's, does not apply to the present document.

12.5 Profile of X.509 certificates for authentication of applications

GEM [1], clause 12.5 is included in the present document.

12.6 Security policy for applications

12.6.1 General principles

GEM [1], clause 12.6.1 is included in the present document with the following modifications:

- In the present document, only MHP codesigning mechanisms apply; alternatives are not permitted.

12.6.2 Permission request file

GEM [1], clause 12.6.2 is included in the present document, with the following notes and modifications.

12.6.2.0 General

GEM [1], clause 12.6.2.0 does not apply to the present document.

12.6.2.1 File encoding

GEM [1], clause 12.6.2.1 is included in the present document.

12.6.2.2 File integrity

GEM [1], clause 12.6.2.2 is included in the present document.

12.6.2.3 Example

GEM [1], clause 12.6.2.3 is included in the present document.

12.6.2.4 Permission request file name and location

GEM [1], clause 12.6.4 is included in the present document, with the following notes and modifications.

Table 44 in GEM [1], clause 12.6.2.4, "Application names for different application types," contains a reference to the name `initial_class_byte`. For the present document, this is to be interpreted as referring to `initial_class_byte` from table 84.

12.6.2.5 Permission request file

GEM [1], clause 12.6.2.5 is included in the present document.

12.6.2.6 Credentials

GEM [1], clause 12.6.2.6 is included in the present document, with the following modifications:

- Support for Credentials is required in the present document.
- Functional equivalents are not permitted.
- Replace the sentence "The certification chain authenticates the grantor's certificate. This chain shall derive from one of the root authorities." with "The certification chain authenticates the grantor's certificate. This chain shall derive from one of the root authorities embedded in the MHP terminal".

12.6.2.7 File Access

GEM [1], clause 12.6.2.7 is included in the present document.

12.6.2.8 CA API

GEM [1], clause 12.6.2.8 is included in the present document, with the following modifications:

- GEM [1], clause 12.6.2.8.0, "GEM introduction," is not included in the present document.

12.6.2.9 Application lifecycle control policy

GEM [1], clause 12.6.2.9 is included in the present document.

12.6.2.10 Return channel access policy

GEM [1], clause 12.6.2.10 is included in the present document.

12.6.2.11 Tuning access policy

GEM [1], clause 12.6.2.11 is included in the present document.

12.6.2.12 Service selection policy

GEM [1], clause 12.6.2.12 is included in the present document.

12.6.2.13 Media API access policy

GEM [1], clause 12.6.2.13 is included in the present document.

12.6.2.14 Inter-application communication policy

GEM [1], clause 12.6.2.14 is included in the present document.

12.6.2.15 User Setting and Preferences access policy

GEM [1], clause 12.6.2.15 is included in the present document.

12.6.2.16 Network permissions

GEM [1], clause 12.6.2.16 is included in the present document.

12.6.2.17 Dripfeed permissions

GEM [1], clause 12.6.2.17 is included in the present document.

12.6.2.18 Privileged Runtime Code Extension Permission

GEM [1], clause 12.6.2.18 is included in the present document with the following notes and modifications.

- NOTE 2: In ECMAscript, code executed via runtime code extension has the same permission level as the enclosing application. Thus, granting an unsigned application the ability to do runtime code extension with string data does not give the application the ability to execute a string as privileged code.

12.6.2.19 Application storage

GEM [1], clause 12.6.2.19 is included in the present document.

12.6.2.20 Non-CA smart card access

GEM [1], clause 12.6.2.20 is included in the present document.

12.6.2.21 Provider Management

GEM [1], clause 12.6.2.21 is included in the present document.

12.6.2.22 Service type selection policy

GEM [1], clause 12.6.2.22 is included in the present document.

12.6.2.23 Privileged application access

GEM [1], clause 12.6.2.23 is included in the present document.

12.6.3 Specific issues for telephone based return channels

MHP terminals with a telephone based return channel shall not connect to any phone number other than that of the default isp without the explicit approval, at least once, of the end-user for the exact phone number concerned. This permits both implementations which always ask the end user and implementations which cache approved phone numbers in a "white list". For DVB-J applications, approval shall be requested when the application calls the `ConnectionRCInterface.connect()` method and failure reported by a `ConnectionFailedEvent`.

If a MHP terminal is going to identify the source of a MHP application to the end-user, it should present to the end-user at least the organization specific text from the OrganisationName field within the Subject Distinguished Name from the leaf certificate used by the terminal to authenticate that application.

12.7 Example of creating an application that can be authenticated

GEM [1], clause 12.7 is included in the present document.

12.8 MHP certification procedures

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

12.9 Certificate management

12.9.1 Certificate Revocation Lists

GEM [1], clause 12.9.1 is included in the present document, with the following notes and modifications:

- Functional equivalents are not permitted.

12.9.2 Root certificate management

GEM [1], clause 12.9.2 is included in the present document, with the following notes and modifications:

- Functional equivalents are not permitted.

12.9.2.1 Introduction

GEM [1], clause 12.9.2.1 is included in the present document, with the following notes and modifications:

- MHP root certificate management is required. Functional equivalents are not permitted.

12.9.2.2 Security of RCMM

GEM [1], clause 12.9.2.2 is included in the present document.

12.9.2.3 Format of RCMM

The encoding of RCMM is ASN.1 DER (see ASN.1 [23]):

```
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 }
```

NOTE: 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 most recent RCMM shall be placed in a file named:

```
"dvb.rcmm"
```

The RCMM files are inserted on a sample basis specified by the CA. These files shall be located in root directories of the object carousels broadcast.

Older RCMMs shall be placed in files named `"dvb.rcmm."<x>` where `x` is a textual representation of an integer decimal number without leading zeroes. The range of values represented in any single directory shall start with 1 and increment in steps of 1.The first unused integer value in the ascending sequence indicates the end of the range. RCMMs shall be sorted in chronological order with the oldest RCMM in `dvb.rcmm.1`. The RCMM signalled as `"dvb.rcmm"` shall not be duplicated in the `"dvb.rcmm."<x>` sequence.

12.9.2.5 RCMM Processing

GEM [1], clause 12.9.2.5 is included in the present document with the following notes and modifications:

The reference to the `thisUpdate` field is to be interpreted as referring to the version of `thisUpdate` from the present document, as specified in clause 12.9.2.3.

The reference to the `nextNbOfSignature` field is to be interpreted as referring to the version of `nextNbOfSignature` from the present document, as specified in clause 12.9.2.3.

12.9.2.6 Example: Renewal of a root certificate

GEM [1], clause 12.9.2.6 is included in the present document.

12.9.3 Test certificates

GEM [1], clause 12.9.3 is included in the present document.

12.10 Security on the return channel

GEM [1], clause 12.10 is included in the present document.

12.11 The internet profile of X.509 (informative)

GEM [1], clause 12.11 is included in the present document.

12.12 Platform minima

GEM [1], clause 12.12 is included in the present document, with the following changes:

- Functional equivalents for authentication mechanisms are not permitted in the present document.
- All platform minima are required in the present document.

12.13 Plug-ins

GEM [1], clause 12.13 is included in the present document.

12.14 Applications loaded from an interaction channel

GEM [1], clause 12.14 is included in the present document.

12.15 Stored and cached applications

GEM [1], clause 12.15 is included in the present document.

12.16 Inner applications and content embedded within other applications

Inner applications shall inherit the set of permissions granted to the application in which they are embedded.

12.17 Authentication of unbound applications

GEM [1], clause 12.17 is included in the present document, with the following changes:

- This clause is required in the present document.

12.18 Authentication of privileged applications

GEM [1], clause 12.18 is included in the present document with the following notes and modifications:

MHP applications which have not been successfully authenticated by such an additional authentication system shall not receive any instances of MonitorAppPermission.

In order to obtain any instances of MonitorAppPermission, an MHP application be; successfully authenticated by such an additional authentication system signalled with an `application_id` in the range for privileged applications GEM [1], table 13, "Value ranges for application_id".

This clause is required in the present document.

13 Graphics reference model

GEM [1], clause 13 is included in the present document, with the notes and modifications detailed in the following clauses.

References to annex G, "Minimum Platform Capabilities" shall be understood to refer to annex G of the present document.

13.1 Introduction

GEM [1], clause 13.1 is included in the present document.

13.2 General Issues

GEM [1], clause 13.2 is included in the present document.

13.3 Graphics

GEM [1], clause 13.3 is included in the present document with the following modifications:

GEM [1], clause 13.3.7 describes the 14:9 aspect ratio as optional for GEM terminals. This clause is required for MHP terminal specifications.

13.4 Video

GEM [1], clause 13.4 is included in the present document with the following modifications:

The NOTE beginning, "GEM does not mandate a particular broadcast streaming format…" is not included in the present document. For a list of formats that must be supported, refer to clause 7.2, "Broadcast streaming formats" of the present document.

13.5 Subtitles

GEM [1], clause 13.5 is included in the present document.

NOTE: This clause applies to all MHP terminals.

13.6 Approximations

GEM [1], clause 13.6 is included in the present document.

14 System integration aspects

14.1 Namespace mapping (DVB Locator)

An extended format of the DAVIC DVB URL (DAVIC 1.4.1p9 [3]) shall be used for addressing DVB-SI entities as well as files within object carousels. This extension of the DAVIC locator is backwards compatible with both the original DAVIC locator as well as the UK DTG extension (UK MHEG Profile). The main extensions are support for multiple component tags for specifying a subset of the components of a service, and a specified way of referencing files in an object carousel within a service.

Using the same informal notation as used above, the following locator formats shall be used:

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

or

```
dvb://'<textual_service_identifier>'[.<component_tag>{&<component_tag>}][;<event_id>]]
     {/<path_segments>}
```

A more formal specification of the DVB URL expressed in BNF (as used in RFC 2396 [18]) is presented below.

Table 101: DVB URL syntax

dvb_url	= dvb_scheme ":" dvb_hier_part
dvb_scheme	= "dvb"
dvb_hier_part	= dvb_net_path | dvb_abs_path
dvb_net_path	= "//" dvb_entity [dvb_abs_path]
dvb_entity	= dvb_transport_stream | dvb_service | dvb_service_component
dvb_transport_stream	= original_network_id "." transport_stream_id
dvb_service	= dvb_service_without_event [dvb_event_constraint]
dvb_service_component	= dvb_service_without_event "." component_tag_set [dvb_event_constraint]
dvb_service_without_event	= original_network_id "." [transport_stream_id] "." service_id | """ textual_service_identifier """
component_tag_set	= component_tag *("&" component_tag)
dvb_event_constraint	= ";" event_id
original_network_id	= hex_string
transport_stream_id	= hex_string
service_id	= hex_string
component_tag	= hex_string
event_id	= hex_string
hex_string	= 1*hex
hex	= digit | "A" | "B" | "C" | "D" | "E" | "F" | "a" | "b" | "c" | "d" | "e" | "f"
digit	= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
dvb_abs_path	= "/" path_segments
NOTE 1: Path_segments as defined in RFC 2396 [18].	
NOTE 2: textual_service_identifier = 1*(unreserved | escaped).	

NOTE: This syntax is fully compliant with the generic syntax of URIs as specified in RFC 2396 [18] and uses the registry-based naming authority version of that. Furthermore, all generic definitions specified in RFC 2396 [18] are valid for the DVB URL as well (e.g. escaping of special characters within file names, etc.).

RFC 2396 [18] defines methods for path segments to include parameters (introduce with a semicolon character ";"). The present document currently makes no use of such parameters. Implementations conforming to the present document 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.

The numeric identifiers original_network_id, transport_stream_id (if present) and service_id shall be matched against the corresponding fields in the SDT.

If present, the network_id shall be matched against the corresponding field in the NIT.

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 the present document.

If present, the network_id shall be matched against the corresponding field in the NIT.

The numeric identifiers original_network_id, transport_stream_id (if present) and service_id shall be matched against the corresponding fields in the SDT.

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 clause 7.1.5, "Monomedia format for text").
- The directory separator character shall be a slash character (0x2F).
- An absolute filename starts with a slash character (as indicated in the BNF above).

14.1.5 dvb_entity = dvb_transport_stream

The numeric identifiers original_network_id and transport_stream_id shall be matched against the corresponding fields in the NIT.

If present, the network_id shall be matched against the corresponding field in the NIT.

14.1.6 textual_service_identifier

The textual_service_identifier in a DVB Locator is encoded in UTF-8. See clause 14.10.1 for the syntax of this field, and clause 14.10.2 for usage.

14.1.7 Application locator

A locator for an application in the current service can be identified by the following specific forms Only applications that are visible in the application database using the current service filter can be found by this locator.

Selecting this locator will launch the application, with the associated parameters.

Table 102

Locator	Meaning
dvb://current.ait/Orgid.Appid?param=val&	An application in the service currently selected by the application.

Where `Orgid` and `Appid` are both zero, this shall define a special locator meaning no applications. If this is the only application locator among a set of locators then no applications shall be launched.

14.2 Reserved names

GEM [1], clause 14.2 is included in the present document.

14.3 XML notation

GEM [1], clause 14.3 is included in the present document, with the following notes and modifications.

In GEM [1], the fourth bullet item under "Rules for encoding of the XML formatted files" reads:

- GEM terminal specifications may extend the allowed XML notation by permitting other character encodings, if this extension is explicitly stated in a GEM terminal specification. If no encoding is specified in an XML file, however, the default shall be UTF-8.

The present document replaces that text with the following:

- The possible XMLDecl item in the beginning of the file shall not contain an "encoding" attribute specifying another encoding than UTF-8.

In other words, MHP prohibits indicating an encoding attribute in an XMLDecl item to specify an encoding other than UTF-8.

14.4 Network signalling

GEM [1], clause 14.4 is included in the present document.

14.5 Text encoding of application identifiers

GEM [1], clause 14.5 is included in the present document, with the following notes and modifications.

The references to `organization_id` and `application_id` are to be interpreted as referring to the versions of `organization_id` and `application_id` from the present document.

14.6 Filename requirements

GEM [1], clause 14.6 is included in the present document with the following notes and modifications.

To GEM [1], clause 14.6.2, "DSMCC object carousel", add the following sentence:

- Other object carousel limitations are found in clause B.2.6, "Mapping of objects to data carousel modules".

14.7 Files and file names

GEM [1], clause 14.7 is included in the present document, with the following notes and modifications.

The reference to "using a locator that refers to a file or directory, as described in clause 14.8" shall be interpreted to mean the use of a DVB locator including the `dvb_abs_path` part of the name part of the syntax.

14.8 Locators and content referencing

The table below lists the types of entity which the present document requires locators to be able to address. It defines the text representation for each entity. Where no text representation is standardized, an implementation dependent representation shall be used. The present document does not require support for addressing any other type of entity in an MHP system by locator or URL.

Table 103: Addressable entities, locators and their text representation

Entity	Text Representation
Transport stream	DVB locator including "dvb_transport_stream" element.
Network	No standardized text representation.
Bouquet	No standardized text representation.
DVB Service	• DVB locator including "dvb_service" element (see note 3). • No standardized text representation for services referenced via `org.dvb.locator.FrequencyLocator`.
Service other than DVB service, e.g. stored application service, abstract service and sub-classes of InternetClientService	No standardized text representation.
DVB Event	DVB locator including "dvb_service" element and "dvb_event_constraint" element.
MPEG Elementary Stream	If the elementary stream is signalled with a component tag then this shall be a DVB locator including the "dvb_service_component" element otherwise there is no standardized text representation.
File	"file:", "http:" and "https:" URLs, as referred to in GEM [1], clause 14.8.
Directory	"file:", "http:" and "https:" URLs, as referred to in GEM [1], clause 14.8.
Drip feed decoder	"dripfeed://"
MHP application	See clause 14.1.7, "Application locator".
NOTE 1: The hostname part of a "file:" URL shall always be the empty string. These URLs are always in the namespace of the receiver. Transmitting them across a network as part of an application is meaningless. NOTE 2: DVB locators including the "dvb_abs_path" element may be returned by MHP APIs as a mechanism to provide references to files in carousels which may not currently be mounted. These locators can only be used to mount new carousels or be translated into a "file:" URL once a new carousel has been mounted. NOTE 3: It is not recommended to include the transport_stream_id in references to DVB services because the pair (original_network_id, service_id) already uniquely identifies the service.	

The DVB specifications define two places where multiple logical service components can be carried in a single MPEG elementary stream - DVB subtitles and MPEG-2 multichannel audio. The present document does not provide locators to distinguish between multiple languages of subtitles or audio carried in a single MPEG elementary stream in this way. Hence methods returning locators for service components / elementary streams shall return the same locator regardless of any selection between such logical service components. Methods accepting locators for service components / elementary streams shall select between any such logical service components based on the rules for elementary stream selection in GEM [1], clause 11.4.2.2 "Clarifications".

Where there is no standardized text representation, the implementation specific representation can be used to construct Locators which address the original addressable entity but only within that single application instance.

NOTE: This means there is no requirement to be able to re-use implementation specific representations after reading them in from persistent storage or over the inter-application communication API.

14.9 Content referencing for IPTV

GEM [1], clause 14.9 is included in the present document.

14.10 Service identification

In the present document, there are two mechanisms for uniquely identifying a service:

- The triplet of numeric SI identifiers:

  `original_network_id`, `transport_stream_id` and `service_id` (corresponding to identifiers with the same name defined by EN 300 468 [4] carried in the SI of the broadcast)

NOTE: It is not recommended to use the `transport_stream_id` because the tuple `original_network_id`, `service_id` already uniquely identifies a service.

- A textual service identifier:

  textual_service_identifier_bytes carried in the optional Service identifier descriptor in the SDT (see clause 10.12.1, "Service identifier descriptor").

Both enable global, unique identification of a service.

The textual identifier has additional properties:

- They can identify two (or more) service instances as being the same service even if they for technical reasons have different numeric identifiers.

  It is up to the service provider to decide whether different service instances are identified as being the same service.

- They can give alternative identifications for a single service.

14.10.1 Syntax of the textual service identifier

The syntax of the textual service identifier is:

```
<service_name> "." <service_provider_domain_name>
```

where:

<service_name>: is a unique name for the service within the service provider's domain

<service_provider_domain_name>: is an Internet DNS domain name that the service provider has rights to control. The organization's administrating the Internet DNS domain names are used as a globally unique registration mechanism that allows these textual service identifiers to be globally unique names.

The <service_name> field shall follow the rules defined for Internet DNS names so that the whole textual service identifier is a valid host name to be used in the Internet DNS as defined in RFC 1035 [27].

An example of a textual service identifier is:

```
movie-channel-1.broadcaster-b.com
```

where `"broadcaster_b.com"` is an Internet DNS domain owned by the broadcaster and `"movie_channel_1"` is a unique name for the service assigned by the service provider

NOTE: The textual service identifier has the same syntax as an Internet host name and it has to be assigned in a domain that the service provider has the rights to control. However, the textual service name for a service is not required to resolve to any IP address using the Internet DNS service and if it does, this version of the present document does not specify any specific services that this host should provide if contacted using the IP protocols.

14.10.2 Handling of the textual service identifiers within the MHP terminal

The MHP terminal discovers the textual service identifiers for a given service from the SDT table similarly as it discovers the existence of the service in the first place. The MHP terminal shall know the textual service identifiers for the available services in the same way that it knows the numeric identifiers: `original_network_id`, `transport_stream_id` and `service_id`.

When the application uses a URI referring to a DVB service, the resolution of this URI to the necessary information needed to locate the service happens in the same way regardless of if this URI contains a textual identifier or the numeric identifiers.

The URI string provided by the application shall be considered to match the one included in the SDT when the strings are the same.

14.11 CA system

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

If a functioning CA system is exposed to MHP applications at all then it shall be equally exposed through all CA related API features, e.g.:

- The CA API.
- CA related reason codes of `org.dvb.media.PresentationChangedEvent`.
- `javax.tv.service.selection.PresentationChangedEvent` and sub-classes.

14.11.1 Service selection

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

In the specific case of the DVB Common Interface this will occur when a Host Control tune request is made by a module. A `javax.tv.service.selection.AlternateContentEvent` shall be sent in this case.

14.11.2 Media component selection

Requests from the CA system to change in composition of the media components (Audio, Video or subtitles) shall be automatically performed (this effect is equivalent effect to using javax.tv.media.MediaSelectControl) and shall be reported to the application by the `org.davic.net.ca.PIDChangeEvent` and `org.dvb.media.PresentationChangedEvent` with reason code either `CA_FAILURE` or `CA_RETURNED` as specified for that event. It shall also be reported by sending `MediaSelectEvents` to all currently registered `MediaSelectListeners` whose current selection would be changed by this.

In the specific case of the DVB Common Interface this will occur when a Host Control replace / clear_replace request is made by a module. Events `javax.tv.service.selection.AlternativeContentEvent` and `NormalContentEvent` shall be sent respectively.

14.11.3 Non-media component selection

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

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

14.12 Focus management

GEM [1], clause 14.12 is included in the present document.

15 Detailed platform profile definitions

This clause 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. The Enhanced Broadcast profile and the Interactive Broadcast profile mirror two of the profiles in GEM [1].

Table 104: Detailed platform profile definitions

Area	Specification	Enhanced Broadcast Profile 3	Interactive Broadcast Profile 3	Internet Access Profile 3	IPTV Profile 3
Broadcast channel protocols					
	6.2.2, "MPEG-2 Sections"	M	M	M	M
	6.2.5, "DSM-CC User-to-User Object Carousel"	M	M	M	M
	IP Multicast stack based on: 6.2.6, "DVB Multiprotocol Encapsulation", 6.2.7, "Internet Protocol (IP)" 6.2.8, "User Datagram Protocol (UDP)" 6.2.10, "IP signalling"	O	Ro	M	Ro
Interaction channel protocols					
TCP/IP	6.3.3, "Transmission Control Protocol (TCP)" 6.3.2, "Internet Protocol (IP)"	-	M	M	M
UDP/IP	6.3.2, "Internet Protocol (IP)" 6.3.9, "User Datagram Protocol (UDP)"	-	M	M	M
DSM-CC U-U RPC	6.3.4, "UNO-RPC" 6.3.5, "UNO-CDR" 6.3.6, "DCM-CC User to User"	-	O	O	O
HTTP	6.3.7.1, "HTTP 1.1"	-	O	O	O
HTTP	6.3.7.2, "MHP profile of HTTP 1.0"	-	M	M	M
DNS	6.3.10, "DNS"	-	M	M	M
HTTPS	6.3.7.3, "HTTPS"	-	M	M	M
Interaction Channel File System	6.4.1, "File system implemented only via the interaction channel"	-	M	M	M
DSMCC / HTTP hybrid	6.4.2, "Hybrid between broadcast stream and interaction channel"	-	M	M	M
IPTV	6.5.1.1, "Service Discovery and Selection"	-	-	-	M
	6.5.1.2, "Broadband Content Guide"	-	-	-	O (note 3)
	6.5.1.3, "RTP"	-	-	-	O
	6.5.1.4, "RTSP"	-	-	-	M
	6.5.2.1, "IP service discovery"	-	-	-	M
	6.5.2.2, "Broadband content guide"	-	-	-	O (note 3)
Static formats					
Bitmap pictures	7.1.1.3, "PNG" + 15.1, "PNG - restrictions"	M	M	M	M
	7.1.1.3, "PNG" without restrictions	n/r	n/r	n/r (note 2)	n/r
	7.1.1.4, "GIF"	n/r	n/r	n/r (note 2)	n/r
	7.1.2, "MPEG-2 I-Frames"	M	M	M	M
	7.1.1.2, "JPEG" + 15.3, "JPEG - restrictions"	M	M	M	M
	7.1.1.2, "JPEG"	-	M	M	M
Audio clips	7.1.4, "Monomedia format for audio clips"	M	M	M	M
Video drips	7.1.3, "MPEG-2 Video "drips""	M	M	M	M

Area	Specification	Enhanced Broadcast Profile 3	Interactive Broadcast Profile 3	Internet Access Profile 3	IPTV Profile 3
Text encoding	7.1.5, "Monomedia format for text"	M	M	M	M
Broadcast streaming formats					
Video	7.2.2, "Video"	M	M	M	M
Audio	7.2.1, "Audio"	M	M	M	M
Subtitles	7.2.3, "Subtitles"	M	M	M	M
Fonts					
Built in	Character set see annex E, "(normative): Character set", Metrics see annex D, "(normative): Text presentation" Face: UK RNIB "Tiresias"	M	M	M	M
Downloadable	7.4.1, "PFR"	M	M	M	M
	7.4.2, "OpenType"	O	O	O	O
DVB-HTML					
DVB-HTML	8, "DVB-HTML"	n/r	O (note 1)	O (note 1)	O (note 1)
Application model					
Application model	All parts of clause 9, "Application model" except those clauses (and their clauses) identified below.	M	M	M	M
	9.3, "DVB-HTML Model"	-	O (note 1)	O (note 1)	O (note 1)
	9.5, "Life cycle of Xlets embedded in DVB-HTML"	-	O (note 1)	O (note 1)	O (note 1)
	9.6.1, "Applications loaded from the interaction channel"	-	M	M	O
	9.7, "Lifecycle of internet access applications"	-	-	M	O
	9.13, "Unbound applications"	O (note 5)	O (note 5)	O (note 5)	O (note 5)
Application signalling					
Application signalling	All parts of clause 10, "Application Signalling" except those clauses (and their clauses) identified below.	M	M	M	M
	10.6.2.2, "DVB-HTML"	-	O (note 1)	O (note 1)	O (note 1)
	10.8.1.3, "Transport via interaction channel"	-	M	M	O
	10.10, "DVB-HTML Specific descriptors"	-	O (note 1)	O (note 1)	O (note 1)
	10.16.1, "Service bound application signalling"	-	-	-	M
	10.16.2, "XAIT"	-	-	-	O (note 5)
	10.17, "XAIT for classical DVB networks"	O (note 5)	O (note 5)	O (note 5)	O
DVB-J					
DVB-J Platform	All parts of clause 11, "DVB-J Platform" except those clauses (and their clauses) identified below.	M	M	M	M
	11.4.2.10, "Additional and modified semantics for IPTV"	-	-	-	M
	11.5.2, "Support for Multicast IP over the Broadcast Channel"	O	Ro	M	-
	11.5.3, "Support for IP over the Return Channel"	-	M	M	M
	11.6.3.2, "Tuning in IPTV"	-	-	-	M
	11.6.6, "Service discovery and selection for IPTV"	-	-	-	M
	11.6.7, "Integration between protocol independent SI API and TV-Anytime"	-	-	-	O (note 3)
	11.7.10, "Content referencing for IPTV"	-	-	-	M
	11.7.11, "TV-Anytime content referencing and metadata"	-	-	-	O (note 3)
	11.8.2, "APIs for return channel security"	-	M	M	M
	11.8.6, "DVB Extensions for Cryptography"	-	M	M	M
	11.9.5.2, "JDOM"	-	-	-	M

Area	Specification	Enhanced Broadcast Profile 3	Interactive Broadcast Profile 3	Internet Access Profile 3	IPTV Profile 3
	11.9.6, "MHP terminal hardware API"	O (note 4)	O (note 4)	O (note 4)	O (note 4)
	11.11.4.5, "Content referencing for IPTV"	-	-	-	M
	11.12.2.4, "Stored application management API"	O (note 4)	O (note 4)	O (note 4)	O (note 4)
	11.13, "Support for DVB-HTML"	O (note 1)	O (note 1)	O (note 1)	O (note 1)
	11.14, "Internet access"	-	-	M	O
	11.15.2, "OCAP Annex G - the org.ocap.application package"	O (note 4)	O (note 4)	O (note 4)	O (note 4)
	11.15.3, "OCAP Annex P - the org.ocap.service package"	O (note 5)	O (note 5)	O (note 5)	O (note 5)
	11.15.4, "OCAP Annex Q - the org.ocap.system package"	O (note 4)	O (note 4)	O (note 4)	O (note 4)
	11.15.5, "OCAP annex O - the org.ocap package"	O (note 4)	O (note 4)	O (note 4)	O (note 4)
	11.15.6, "OCAP annex U - the org.ocap system.event"	O (note 4)	O (note 4)	O (note 4)	O (note 4)
	17, "Internet access clients"	-	-	M	O

NOTE 1: DVB-HTML is a single option which shall only be considered included if all parts of it are included.
NOTE 2: Recommended to be implemented as part of WWW browser but outside the scope of the MHP conformance regime
NOTE 3: Broadband content guide is a single option that shall only be considered included if all parts of it are included.
NOTE 4: Support for privileged applications is a single option that shall only be considered included if all parts of it are included.
NOTE 5: Support for unbound applications is a single option that shall only be considered included if all parts of it are included. Support for this option is mandatory if privileged applications are supported.

Where an MHP product claims to support a feature defined as optional, recommended optional or not required in the above table, it shall be supported according to the present document, shall be subject to the MHP conformance regime and shall report that the feature is supported using the mechanisms defined in GEM [1], table 41, "System properties for optional feature interrogation".

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

15.1 PNG - restrictions

GEM [1], clause 15.1 is included in the present document.

15.2 Minimum media formats supported by DVB-J APIs

GEM [1], clause 15.2 is included in the present document, with the following notes and modifications.

- Support for subtitles is required in the present document.
- Support of MPEG-2 Video "drips" is required in the present document.
- Functional equivalents are not permitted in the present document.

15.3 JPEG - restrictions

GEM [1], clause 15.3 is included in the present document.

15.4 Locale support

GEM [1], clause 15.4 is included in the present document.

15.5 Video raster format dependencies

This clause addresses the aspects of the present document that vary as a consequence of the video raster format. The formats' names follow those used in TR 101 154 [i.1].

15.5.1 25 Hz standard definition

15.5.1.1 Logical pixel resolution

The logical pixel resolution shall be 72 dots per inch.

15.6 MHP mapping of GEM functional equivalents

Most of the functional equivalents defined in GEM [1] originated as abstractions of capabilities previously defined in MHP. While functional equivalents are not permitted in the present document, a mapping between GEM functional equivalents and their corresponding MHP features may be useful to implementors. Table 105 presents this mapping.

Table 105: MHP mapping of GEM functional equivalents

GEM Functional Equivalent Name	MHP Definition
Arch	Clause 5, "Basic Architecture"
Carousel	Clause 6.2.5, "DSMCC User-to-user Object Carousel"
IP MPE	Clause 6.2.6, "DVB Multiprotocol Encapsulation"
SI	Clause 6.2.9, "DVB Service Information"
	Clause 11.6.1, "DVB Service Information API" Clause 11.11.3, "Bouquet"
	Annex O, "Integration of the JavaTV SI API"
Broadcast IP signalling	Clause 6.2.10, "IP signalling"
IPTV Protocols	Clause 6,5, "IPTV protocols"
Audio	Clause 7.2.1, "Audio"
Video	Clause 7.2.2, "Video"
Subtitles	Clause 7.2.3, "Subtitles"
	Clause 11.4.2.5.1, classes related to subtitles
Audio Clips	Clause 7.1.4, "Monomedia format for audio clips"
Resident Fonts	Clause 7.3
Downloadable Fonts	Clause 7.4
Application Signalling	Clause 10.2, "Program Specific Information" Clause 10.3, "Notation" Clause 10.4, "Application Information Table" Clause 10.5, "Application identification" Clause 10.6, "Control of application life cycle" Clause 10.7, "Generic descriptors" Clause 10.8, "Transport protocol descriptors" Clause 10.9, "DVB-J specific descriptors" Clause 10.11, "Constant values"
	Clause 11.7.2, "Application discovery and launching APIs"
Application Authentication	Clause 12.2, "Authentication of applications"
	Clause 12.9.1, "Certificate Revocation Lists"
Conditional Access	Clause 11.4.2.5.1, classes related to conditional access Clause 11.6.4, "Conditional Access API" Clause 14.10, "CA system"
Content Referencing	Clause 11.7.6, "Content Referencing"
	Clause 11.11.11, "Methods working on many Locator types"
	Clause 14.1, "Namespace mapping (DVB Locator)"
	Clause 14.9, "Service identification"
Graphics Resolution	Clause D.3.4.2, "Horizontal resolution"
	Clause G.1.1, "Device capabilities"
	Clause G.4, "Resident fonts and text rendering"
Text Wrapping	Clause D.3.7.2, "Text wrapping setting is true"
RCMM	Clause 12.9.2, "Root Certificate Management"
Active Format Descriptor	(no MHP definition)

16 Registry of Constants

GEM [1], clause 16 is included in the present document with the following modifications.

16.1 System constants

GEM [1], clause 16.1 is included in the present document with the following modifications.

The present document adds the following table listing profile encodings.

Table 106: Profile encoding

Application profile	Version			Definition
	Major	Minor	Micro	
1	1	0	3	Enhanced Broadcast Profile 1 as defined in the present document
2	1	0	3	Interactive Broadcast Profile 1 as defined in the present document
1	1	1	3	Enhanced Broadcast Profile 2 as defined in the present document
2	1	1	3	Interactive Broadcast Profile 2 as defined in the present document
3	1	1	3	Internet Access Profile 2 as defined in the present document
1	1	2	0	Enhanced Broadcast Profile 3 as defined in the present document
2	1	2	0	Interactive Broadcast Profile 3 as defined in the present document
3	1	2	0	Internet Access Profile 3 as defined in the present document
4	1	2	0	IPTV Profile 3 as defined in the present document

16.2 DVB-J constants

GEM [1], clause 16.2 is included in the present document.

17 Internet access clients

GEM [1], clause 17 is included in the present document with the following changes:

This feature is required in MHP.

MHP internet access clients which support a particular element.attribute from table 6, "Use of ContentType attribute on elements" and table 8, "Use of MIME media type by element" to a specific MIME type listed in the table shall support the DVB-HTML behaviour defined for that type of reference. This applies to the following MIME types only:

- audio/mpeg, video/mpeg.
- multipart/dvb.service.

Annex A (normative): External references; errata, clarifications and exemptions

GEM [1], annex A is included in the present document with the following notes and modifications.

A.1 Errata to GEM

Void.

A.2 Errata to DAVIC

The following errata to DAVIC [3] shall apply.

A.2.1 org.davic.mpeg.dvb

A.2.1.1 General

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

- `DvbService;`
- `DvbElementaryStream;`
- `DvbTransportStream.`

A.2.2 org.davic.net

A.2.2.1 ca

A.2.2.1.1 CAMessage

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

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

This class represents messages to CA modules.

Constructors:

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

Constructor for the message.

Parameters:

data - message data

Method Detail:

getData

public byte[] getData()

Returns:

the data of the message

A.2.2.1.2 CAModule

A.2.2.1.2.1 buyEntitlement(org.davic.net.Locator)

This method is considered to have the amendment.

Replace:

> Initiates a purchase dialogue for specified service or future event (specified by a Locator).

With:

> Request to buy a specified service or future event (specified by a Locator) from a conditional access system.

In the comments of `org.davic.net.ca.CAModule.buyEntitlement()` the sentence:

> In case of CA0 this maps onto event_query with event_cmd_id = mmi (Common Interface specification, clause B.4.1.1).

is replaced with

> In case of DVB Common Interface, this maps onto CI messages as follows:
>
> - When the Locator points to a service and the terminal is currently receiving the transport stream that this service is carried in and this transport stream is available to this CA module, then this method is mapped to a ca_pmt message with ca_pmt_cmd_id set to "ok_mmi". The value returned in the ca_pmt_reply is mapped as defined in the documentation of the constants in the class. If the module is currently descrambling the service and the terminal is aware of this, ENTITLEMENT_AVAILABLE shall be returned immediately without communicating with the module.
> - When the Locator points to a service that is not carried in a currently received transport stream, NotTunedException is thrown.
> - When the Locator points to an event, this maps onto event_query message with event_cmd_id set to "mmi" (Common Interface specification, clause B.4.1.1). The value returned in the event_reply message is mapped as defined in the documentation of the constants in this class.

In the CA API, the constants defined in the `org.davic.net.ca.CAModule` class are mapped to the `CA_enable` values of the `ca_pmt_response` message and the `event_status` values of the `event_reply` message in the Common Interface protocol as follows:

> ENTITLEMENT_AVAILABLE:
>
> CA_enable value "Descrambling possible" (0x01).
>
> event_status value "entitlement_available" (0x01).
>
> event_status value "mmi_complete_available" (0x05).
>
> ENTITLEMENT_NOT_AVAILABLE:
>
> CA_enable value "Descrambling not possible (because no entitlement)" (0x04).
>
> event_status value "entitlement_not_available" (0x02).
>
> event_status value "mmi_complete_not_available" (0x06).
>
> ENTITLEMENT_UNKNOWN:
>
> CA_enable value "Descrambling not possible (for technical reasons)" (0x05).
>
> all other CA_enable values not having an explicit mapping in this section.
>
> event_status value "entitlement_unknown" (0x00).

event_status value "mmi_complete_unknown" (0x04).

all other event_status values not having an explicit mapping in this section.

MMI_DIALOGUE_REQUIRED:

CA_enable value "Descrambling possible under conditions (purchase dialogue)" (0x02).

CA_enable value "Descrambling possible under conditions (technical dialogue)" (0x03).

event_status value "mmi_dialogue_required" (0x03).

A.2.2.1.2.2 isDescramblable(ElementaryStream streams[])

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

If an empty array is passed in, returns true.

Is considered to have the following text added as a parameters clause:

Parameters:

streams - the set of elementary streams to be tested for the possibility to descramble.

A.2.2.1.2.3 openMessageSession(MessageListener)

This method is considered to have the amendment:

Modify:

Throws: ModuleBusyException.

raised if the module is busy and is not able to handle a message session at the moment.

To say:

Throws: ModuleBusyException.

raised if the module is busy and is not able to handle a message session at the moment. This is CA system dependant.

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

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

NOTE 1: The document referred to as "common interface extensions specification" is TS 101 699 [55].

The following text:

Throws: ModuleResourceNonExistentException

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

Is considered to be replaced by:

Throws: ModuleResourceNonExistentException.

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

NOTE 2: The `ModuleResourceVersionTooLowException` is never be thrown in the present document.

A.2.2.1.2.4 queryEntitlement(org.davic.net.Locator)

This method is considered to include the following:

> Throws: org.davic.net.InvalidLocatorException.
>
> if the locator does not point to a valid service or event.

In the comments of `org.davic.net.ca.CAModule.queryEntitlement()` the sentence:

> In case of CA0 this maps onto event_query with event_cmd_id = query (Common Interface specification, clause B.4.1.1).

is replaced with:

> In case of DVB Common Interface, this maps onto CI messages as follows:
>
> - When the Locator points to a service and the terminal is currently receiving the transport stream that this service is carried in and this transport stream is available to this CA module, then this method is mapped to a ca_pmt message with ca_pmt_cmd_id set to "query". The value returned in the ca_pmt_reply is mapped as defined in the documentation of the constants in the class. If the module is currently descrambling the service and the terminal is aware of this, ENTITLEMENT_AVAILABLE shall be returned immediately without communicating with the module.
> - When the Locator points to a service that is not carried in a currently received transport stream, ENTITLEMENT_UNKNOWN shall be returned.
> - When the Locator points to an event, this maps onto event_query with event_cmd_id = query (Common Interface specification, clause B.4.1.1). The return values is mapped as defined in the documentation of the constants in this class.

A.2.2.1.2.5 sendToModule

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

> In systems based on the DVB common interface, messages sent using this method shall be mapped onto the CAPipelineRequest as defined in clause 6.8.3 of the common interface extensions specification. Responses from the CA system reported through the CAPipelineResponse and CAPipelineNotification messages shall be mapped onto instances of ModuleResponseEvent.

NOTE: The document referred to as "common interface extensions specification" is TS 101 699 [55].

A.2.2.1.2.6 Protected constructor

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

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

A.2.2.1.2.7 Additional methods

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

getApplicationTitle

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

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

* @exception ModuleUnavailableException raised if the physical CA module
*            has been removed and is not available any more
* @return the application title string
```

```
*/
public String getApplicationTitle() throws ModuleUnavailableException;
```

enterApplication

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

This functionality is provided at low lowel (within MMI and Application Information
sessions defined in EN50221) by the message enter_menu() message (Application Information
resource, see 8.4.2.3)

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

closeMMI

```
/**
* Requests the module to leave
* the complete tree of the current high-level MMI dialogs.

This functionality is provided at low lowel (within MMI and Application Information
sessions defined in EN50221) by the message close_mmi() message (MMI resource, see 8.6.2.1)

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

A.2.2.1.2.8 listEntitlements

This method is considered to have the following text added:

> In case of DVB Common Interface, it is not possible to obtain this information from the CI module. Thus this method shall return an array of length zero.

A.2.2.1.3 CAModuleManager

A.2.2.1.3.1 addMMIListener()

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

> If an application has registered (and not removed) a listener to handle the MMI dialogues and if an MMI dialogue is required, this causes the platform to ask the MMI listener to handle the MMI dialogues. If there is no application registered to handle the MMI dialogues, these will be handled by the platform.

A.2.2.1.3.2 getModules(Services)

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

> If the service passed as a parameter is not scrambled, returns an empty array whose length is 0.

A.2.2.1.4 NoFreeCapacityException

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

org.davic.net.ca

NoFreeCapacityException

Syntax

```
public class NoFreeCapacityException extends org.davic.net.ca.CAException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.davic.net.ca.CAException
                    |
                    +--org.davic.net.ca.NoFreeCapacityException
```

All Implemented Interfaces:

java.io.Serializable.

Description:

This exception is thrown when a method is called and the CA module does not have the required capacity to perform the action.

Constructors:

NoFreeCapacityException()

```
public NoFreeCapacityException()
```

Default constructor for the exception

NoFreeCapacityException(String)

```
public NoFreeCapacityException(java.lang.String reason)
```

Constructor for the exception with a specified reason.

Parameters:

`reason` - the reason why the exception was raised.

A.2.2.1.5 MMIObject

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

org.davic.net.ca

MMIObject

Syntax

```
public class MMIObject
```

```
java.lang.Object
  |
  +--org.davic.net.ca.MMIObject
```

Direct Known Subclasses:

List, Text

Description

The base class of all MMI classes.

Methods:

close()

```
public void close()
```

Closes the MMI object and informs the CA API implementation that the application intends to close or has closed the corresponding MMI screen.

A.2.2.1.6 DescramblerProxy

A.2.2.1.6.1 startDescrambling()

In the comments of `org.davic.net.ca.DescramblerProxy.startDescrambling()` (all signature versions) the sentence:

> This method may start an MMI dialog.

is replaced with:

> This method may result in the CA system requesting an MMI dialog.

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

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

A.2.2.1.6.2 startDescrambling(org.davic.mpeg.Service, java.lang.Object)

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

> DescramblerProxy applies from the point of view of one application. Methods such as `startDescrambling()` and `stopDescrambling` apply on a per-application basis and do not impact descrambling on behalf of other applications, except subject to platform resource limitations.

A.2.2.1.6.3 startDescrambling(org.davic.mpeg.ElementaryStream[], java.lang.Object)

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

> If `org.davic.mpeg.ElementaryStream[]` is a zero length array the method has no effect.

A.2.2.1.6.4 startDescrambling(org.davic.mpeg.ElementaryStream[], CAModule, java.lang.Object)

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

> If `org.davic.mpeg.ElementaryStream[]` is a zero length array the method has no effect.

A.2.2.1.6.5 startDescramblingDialog(org.davic.mpeg.ElementaryStream[])

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

> If `org.davic.mpeg.ElementaryStream[]` is a zero length array the method has no effect.

A.2.2.1.6.6 stopDescrambling()

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

> If no descrambling is being done then this method has no effect.

A.2.2.1.6.7 stopDescrambling(org.davic.mpeg.ElementaryStream[] streams)

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

> The method `stopDescrambling( ElementaryStream[])` only stops the descrambling of streams which have been started through this `DescramblerProxy` instance, and not started through any other instance.

Is considered to include the following parameter specification:

streams

Array of ElementaryStreams whose descrambling is to be stopped.

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

> The `stopDescrambling` method only affects members of the array of streams that are being descrambled. There is no effect on any streams listed in the array that are not being descrambled.

A.2.2.1.6.8 startDescramblingDialog

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

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

A.2.2.1.6.9 Class Description

The methods `startDescrambling` and `stopDescrambling` in `DescramblerProxy` are used to add and remove, respectively, elementary streams of one service from the set of elementary streams to be descrambled. The MHP implementation needs to keep track of the set of elementary streams that are requested to be descrambled by the application.

When the application calls `startDescrambling` or `stopDescrambling` and this set of elementary streams changes while the service is being descrambled, an implementation using the Common Interface shall send the CA_PMT message with the ca_pmt_list_management set to "update" and the version number appropriately changed. This CA_PMT message shall include information about only those elementary streams that are to be descrambled. Those elementary streams that are not requested to be descrambled should be omitted from the CA_PMT message completely.

NOTE 1: The ca_pmt_cmd_id can not be used to tag elementary streams that should not be descrambled because there is no suitable value of ca_pmt_cmd_id for this purpose. Therefore these streams need to be left out from the CA_PMT completely.

NOTE 2: Updating the CA_PMT in response to the method calls from the MHP application requires also changing the CA_PMT version_number field. This implies that the MHP implementation needs to have an independent version numbering for the CA_PMTs while descrambling one service and the version_number field of the PMT in the broadcast can not be simply copied into the CA_PMT.

A.2.2.1.7 StartMMIEvent(MMIObject, int, java.lang.Object)

Is considered to include the following parameter specification:

caModule

The CAModule object that is the source of the event, which shall be returned by the getSource() method.

A.2.2.1.8 ModuleResponseEvent

A.2.2.1.8.1 Protected Constructor

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

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

A.2.2.1.9 NewModuleEvent

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

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

A.2.2.1.10 PIDChangeEvent

The following text:

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

is considered to be replaced with the following:

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

NOTE: This event is for information only. The platform is responsible for implementing the requests from the CA system. See also R206-001 [79].

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

The text:

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

and:

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

is considered to be replaced with:

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

and:

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

The text:

```
public Object getSource()
```

Returns the DescramberProxy that is the source of the event.

Overrides:

getSource in class DescramblerEvent.

is considered to be replaced with:

```
public Object getSource()
```

Returns the CAModule that is the source of the event.

Overrides:

getSource in class CAEvent.

A.2.2.1.11 TuneRequestEvent

The following text:

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

is considered to be replaced with the following:

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

NOTE 1: This event is only guaranteed to be delivered to applications that survive the service selection caused by the Host Control tune request (see clause 11.6.4, "Conditional Access API" on page 301).

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

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

The text:

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

and:

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

is considered to be replaced with:

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

and:

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

The text:

```
public Object getSource()
```

Returns the DescramberProxy that is the source of the event.

Overrides:

getSource in class DescramblerEvent.

is considered to be replaced with:

```
public Object getSource()
```

Returns the CAModule that is the source of the event.

Overrides:

getSource in class CAEvent.

A.2.2.1.12 DescramblingStartedEvent

In the case of DVB common interface, this event is sent when the MHP terminal has requested the module to start the descrambling.

A.2.2.1.13 DescramblingStoppedEvent

In the case of DVB common interface, this event is sent when the MHP terminal has requested the module to stop the descrambling.

A.2.2.2 dvb.DvbLocator(int onid, int tsid, int serviceid, int eventid, int componenttags[], String filePath)

The following parameter specification:

the

file path string including the slash character in the beginning.

is considered to read:

filePath

string including the slash character in the beginning.

A.2.2.3 dvb.DvbLocator

A.2.2.3.1 DvbLocator(int, int, int, int, int[])

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

A.2.2.3.2 Additional method

The following method is considered to be present:

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

A.2.2.3.3 getOriginalNetworkId

The returns clause of the `getOriginalNetworkId()` method is considered to include "-1 if not present" similarly as the other methods returning the numeric identifiers.

A.3 CSS 2

With reference to CSS 2 [39].

The present document is considered to include:

11.1.2 Clipping: the "clip" property

A clipping region defines what portion of an element's rendered content is visible. By default, the clipping region has the same size and shape as the element's box(es). However, the clipping region may be modified by the "clip" property.

"clip"

- Value: <shape> | auto | inherit.
- Initial: auto.
- Applies to: block-level and replaced elements.

- Inherited: no.
- Percentages: N/A.
- Media: visual.

The "clip" property applies to elements that have a "overflow" property with a value other than "visible". Values have the following meanings:

- auto

The clipping region has the same size and location as the element's box(es).

- <shape>

In CSS2, the only valid <shape> value is: rect (<top> <right> <bottom> <left>) where <top>, <bottom> <right>, and <left> specify offsets from the respective sides of the box.

<top>, <right>, <bottom>, and <left> may either have a <length> value or "auto". Negative lengths are permitted. The value "auto" means that a given edge of the clipping region will be the same as the edge of the element's generated box (i.e. "auto" means the same as "0").

When coordinates are rounded to pixel coordinates, care should be taken that no pixels remain visible when <left> + <right> is equal to the element's width (or <top> + <bottom> equals the element's height), and conversely that no pixels remain hidden when these values are 0.

The element's ancestors may also have clipping regions (in case their "overflow" property is not "visible"); what is rendered is the intersection of the various clipping regions.

If the clipping region exceeds the bounds of the UA's document window, content may be clipped to that window by the native operating environment.

EXAMPLE(S): The following two rules:

- P { clip: rect(5px, 10px, 10px, 5px); }
- P { clip: rect(5px, -5px, 10px, 5px); }

will create the rectangular clipping regions delimited by the dashed lines in the following illustrations:

- [D].

NOTE: In CSS2, all clipping regions are rectangular. We anticipate future extensions to permit non-rectangular clipping.

While CSS2 specifies that values of "rect()" specify offsets from the respective sides of the box, current implementations interpret values with respect to the top and left edges for all four values (top, right, bottom, and left). The Working Group proposes to revise CSS2 to conform to current practice.

Annex B (normative): Object carousel

B.1 Introduction

The broadcast applications are transmitted using the DSM-CC User-to-User Object Carousels.

The present document is based on the following specifications:

- ISO/IEC 13818-1 [15] - MPEG 2 systems.
- ISO/IEC 13818-6 [16] - DSM-CC.
- EN 301 192 [5] - DVB specification for data broadcasting.
- TR 101 202 [i.6] - Implementation Guidelines for Data Broadcasting.

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 clause, 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 the present document.

For the object carousel messages, also the message syntax is included.

NOTE: In the syntax tables grey shading indicates parts that the broadcaster may put in, but an MHP terminal compliant with the present document 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 clause 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 the present document, 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).	The present document
last_section_number	For sections transporting DownloadDataBlock fragments: - all modules intended to be retrieved shall have the last section number £ 0xFE; - if last section number = 0xFF receiver behaviour is undefined.	

The maximum section length is 4 096 bytes for all types of sections used in Object Carousels. The section overhead is 12 bytes, leaving a maximum of 4 084 bytes of payload per section.

B.2.1.1 Sections per TS packet

Parts of no more than four sections shall be delivered in a single TS packet.

B.2.2 Data Carousel

This clause defines the content of the data carousel messages when used in the object carousel.

Usage of data carousel descriptors not listed below in an MHP object carousel is not defined by the present document. They should not be used by broadcasters and receivers should ignore them.

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 clause B.2.5 "Assignment and use of transactionId values"	The present document
AdaptationLength	The MHP terminal may ignore the possible contents of the dsmccAdaptationHeader field.	

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 4 066 (max. section payload - DDB-header size (18)) The recommended blockSize is 4 066.	DSM-CC (the present document 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 each with the value of 0xFF	DVB / the present document

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 the present document. 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 / The present document
moduleTimeOut blockTimeOut minBlockTime	These fields are defined in units of µs. An appropriate value must be explicitly encoded by carousel generation equipment. There is no default value that may be encoded, i.e. 0xFFFFFFFF has no special meaning. Receivers shall not employ an inbuilt default instead of the signalled value, as there is no way to define these without knowledge of the construction of a particular carousel.	The present document

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++) {				Possible additional taps that may be ignored by MHP terminals.
id	16	uimsbf	+	
use	16	uimsbf	+	
assocTag	16	uimsbf	+	
selector_length	8	uimsbf	N2	
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 clause 10.8.3.2, "Pre-fetch descriptor").

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++) {				
label_byte	8	uimsbf		The label
}				
}				

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

label_byte: These 8-bit fields carry an array of bytes that are a module label. This label matches a label used in one of the pre-fetch descriptors clause 10.8.3.2, "Pre-fetch descriptor". The match shall be done as a byte-by-byte comparison between the two byte arrays.

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 clause B.5.2, "Transparency levels of caching") 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 GEM [1], in table 54, "Transparency level values". The semantics of the policies are defined in GEM [1], clause B.5.2, "Transparency levels of caching".

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.10: Restrictions on the ServiceGatewayInfo

Field	Restrictions	Source
BIOP::ServiceGatewayInfo::downloadTaps	The MHP terminal may ignore the downloadTap list.	The present document
BIOP::ServiceGatewayInfo::serviceContextList	The MHP terminal may ignore the service context list.	
BIOP::ServiceGatewayInfo::UserInfo	The MHP terminal may ignore the user info.	

Table B.11: ServiceGatewayInfo() syntax

Syntax	bits	Type	Value	Comment
ServiceGatewayInfo(){				
IOP::IOR()			+	See table B.22 "IOP::IOR syntax"
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.2.2.6 Download Cancel

There is no semantic for this message in this profile. Receivers may ignore them.

B.2.2.7 DownloadDataBlock

Table B.12: Restrictions on the DDB

Field	Restrictions	Source
moduleId	Module ids are unique within the scope of the object carousel. See ISO/IEC 13818-6 [16], clause 11.2.3.	DSM-CC

B.2.3 The Object Carousel

B.2.3.1 BIOP Generic Object Message

The BIOP Generic Object Message is a common structure used by all the BIOP (Broadcast Inter-ORB Protocol) messages.

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

Field	Restrictions	Source
MessageHeader::byte_order	0 (indicating big-endian byte order)	DVB
MessageSubHeader::objectKey	Maximum length of the key shall be four bytes.	DVB
MessageSubHeader::objectKind	The short three-letter aliases shall be used, plus the null-terminator.	DVB
Access attributes	Access attributes are not transmitted in object carousels	DSM-CC

B.2.3.2 CORBA strings

In a number of places Object Carousel messages include text strings. These are formatted in accordance with clause 12.3.2 of CORBA/IIOP [2] and using the so-called "CDR-Lite" encoding as described by ISO/IEC 13818-6 [16], clause 5.6.3.4. 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 summarized in table B.14.

Table B.14: Location of CORBA format strings

string	Length field size (bits)	location
objectKind_data	32	Table B.16, "BIOP::FileMessage syntax"
objectKind_data id_data kind_data	32 8 8	Table B.19, "BIOP::DirectoryMessage syntax"
objectKind_data	32	Table B.28, "BIOP::StreamMessage syntax"
objectKind_data eventName_data	32 8	Table B.30, "BIOP::StreamEventMessage syntax"
type_id_byte	32	Table B.21, "IOP::IOR syntax"
id_data kind_data	32 32	Table B.25, "Syntax of Lite Options Profile Body with ServiceLocation component"

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 ObjectInfo may be empty (have a length of zero). If not empty the first 8 bytes of the ObjectInfo shall contain the DSM::File::ContentSize attribute. This is optionally followed by a loop of descriptors. The descriptors defined for possible use in this location are: Content type descriptor	The present document
MessageSubHeader::ServiceContextList	The MHP terminal may skip the possible serviceContextList structures.	

Table B.16: BIOP::FileMessage syntax

Syntax	bits	Type	Value	Comment
BIOP::FileMessage() {				
magic	4 x 8	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	1 to 4
for (i=0; i<N1; i++) {				
objectKey_data	8	uimsbf	+	
}				
objectKind_length	32	uimsbf	0x00000004	
objectKind_data	4 x 8	uimsbf	0x66696C00	"fil" type_id alias
objectInfo_length	16	uimsbf	N2	
DSM::File::ContentSize	64	uimsbf	+	objectInfo (see note)
for (i=0; i<N2 - 8; i++) {				
descriptor()	8	uimsbf	+	
}				
serviceContextList_count	8	uimsbf	N3	serviceContextList
for (i=0; i<N3; i++) {				
context_id	32	uimsbf		
context_data_length	16	uimsbf	N4	
for (j=0; j<N4; j++) {				
context_data_byte	8	uimsbf	+	
}				
}				
messageBody_length	32	uimsbf	*	
content_length	32	uimsbf	N5	
for (i=0; i<N5; i++) {				
content_byte	8	uimsbf	+	actual file content
}				
}				
NOTE: If present and non-zero, this shall be the same as the content_length of the referenced FileMessage.				

B.2.3.4 Content type descriptor

Zero or one content type descriptors can be carried in the file `MessageSubHeader::ObjectInfo` or the `BIOP::Binding::ObjectInfo`. Where more than one content type descriptor is used they shall express the same content format. Also, the content type (if any) signalled in the directory binding shall be identical to that signalled in the bound file's header. This optional descriptor identifies the media type of the file.

This content type signalling only applies to objects of type file and is not appropriate for other object types.

If this descriptor is absent or not sufficient to categorize the content type then the extension portion of the file name shall be used be used to provide the media type mapping via GEM [1], table 7, "File type identification". The extension portion of the filename shall not be used if this descriptor is provided.

The format of the content type descriptor is shown in table B.17.

Table B.17: Content type descriptor syntax

Syntax	bits	Type	Value	Comment
content_type_descriptor() {				
descriptor_tag	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 [25] and hence `content_type_data` carries the payload of the Content-Type header defined in RFC 2045 [25].

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.	The present document
MessageSubHeader::ServiceContextList	The MHP terminal may skip the N3 possible serviceContextList structures.	The present document
BIOP::Name	The name shall contain exactly one NameComponent. The id_length shall be 2 or greater. The id_data shall not be replicated for other name components within this directory.	The present document
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	The ObjectInfo for bound objects may be empty (have a length of zero). If the bound object is a file and the ObjectInfo is not empty the first 8 bytes of the ObjectInfo shall 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	The present document

Table B.19: BIOP::DirectoryMessage syntax

Syntax	bits	Type	Value	Comment
BIOP::DirectoryMessage() {				
magic	4 x 8	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	1 to 4
for (i=0; i<N1; i++) {				
objectKey_data	8	uimsbf	+	
}				
objectKind_length	32	uimsbf	0x00000004	
objectKind_data	4 x 8	uimsbf	0x64697200	"dir" type_id alias
objectInfo_length	16	uimsbf	N2 = 0 (note)	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.16, "Restrictions on the BIOP File Message".
for (i=0; i<N6; i++) {				
id_length	8	uimsbf	N7	NameComponent id
for (j=0; j<N7; j++) {				
id_data	8	uimsbf	+	The "/" character shall not be used.
}				
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 [i.6])
}				
}				
}				
BindingType	8	uimsbf	+	0x01 for nobject 0x02 for ncontext
IOP::IOR()			+	objectRef see table B.21, "IOP::IOR syntax"
objectInfo_length	16	uimsbf	N9	
if(kind_data == "fil"){				
DSM::File::ContentSize	64	uimsbf	+	0 means that file size is not signalled
for (j=0; j<N9 - 8; j++) {				
descriptor_byte	8	uimsbf	+	
}				
}				
else {				
for (j=0; j<N9; j++) {				
descriptor_byte	8	uimsbf	+	
}				
}				
}				
}				
NOTE: See item 2 under clause 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".
- Use is made of the service context list.

B.2.3.7 BIOP Interoperable Object References

The Interoperable Object References (IOR) are references to objects and contain the necessary information to locate the object. The IOR structure may contain different options to be able to point to objects that can be reached via different types of connections. For the present document, the use of IORs is limited to references to objects carried in broadcast object carousels. For object carousels, there are two types of object references: one to be used to reference objects carried in the same object carousel and one to be used to reference objects in other object carousels.

Table B.20: Restrictions on the BIOP IOR

Field	Restrictions	Source
IOP::IOR::type_id	Contains the objectKind of the referenced object. A short three-letter aliases shall be used, plus a null-terminator.	The present document
IOP::IOR::taggedProfileList	There shall be at least 1 taggedProfile included in an IOR. For objects carried in a broadcast object carousel, the first taggedProfile shall be either a TAG_BIOP profile or a TAG_LITE_OPTIONS. If the first tagged profile is some other profile, the object is not carried in a broadcast object carousel and the MHP terminal may ignore the object subject to its own capabilities.	The present document

Table B.21: 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.2.3.7.1 BIOPProfileBody

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

Table B.22: Restrictions on the BIOP Profile Body

Field	Restrictions	Source
BiopProfileBody::byte_order	0 (indicating big-endian byte order)	DVB
BiopProfileBody::LiteComponent	The list shall contain exactly 1 BiopObjectLocation and exactly 1 DSM::ConnBinder as the first two components in that order. The MHP terminal may ignore possible other components in the list.	The present document
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.	The present document
DSM::Tap	In the BIOP_DELIVERY_PARA_USE tap, the id field is not used and may be ignored by the MHP terminal.	The present document
DSM::Tap::timeout	This field is defined in units of μs. An appropriate value must be explicitly encoded by carousel generation equipment. There is no default value that may be encoded, i.e. 0xFFFFFFFF has no special meaning. Receivers shall not employ an in-built default instead of the signalled value, as there is no way to define these without knowledge of the construction of a particular carousel.	The present document

Table B.23: BIOP Profile Body syntax

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	1 to 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
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.2.3.7.2 LiteOptionsProfileBody

The LiteOptionsProfileBody is used for making links to objects carried in other object carousels.The LiteOptionsProfileBody can be used to make references to objects carried in other carousels within the same Transport Streams or in other Transport Streams. The following constraints are put on the use of the LiteOptionsProfileBody:

- LiteOptionsProfileBody references shall never be used in an IOR which is in the DSI referencing the service gateway.

- The target carousel is never mounted automatically by the implementation, but the application may do so using the MHP DSMCC API and where necessary, the tuning API.

- When the LiteOptionsProfileBody is encountered the application will get a ServiceXFRErrorEvent or a ServiceXFRException unless the object carousel which is the target of the lite options profile body reference is already mounted by the MHP terminal. In this latter case, the access to the object shall succeed unless the object is unavailable, e.g. the target file does not exist in the target carousel or the target carousel has lost its connection.

Table B.24: Restrictions on the Lite Options Profile Body

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.	The present document
DSM::ServiceLocation	For objects carried in the broadcast object carousel, the service domain NSAP address shall follow the Carousel NSAP address format.	The present document
DSM::ServiceLocation::InitialContext	The MHP terminal may ignore the initial context.	The present document

Table B.25: Syntax of Lite Options Profile Body with ServiceLocation component

Syntax	bits	Type	Value	Comment
LiteOptionsProfileBody {				
profileId_tag	32	uimsbf	0x49534F05	TAG_LITE_OPTIONS (Lite Options Profile Body)
profile_data_length	32	uimsbf	*	
profile_data_byte_order	8	uimsbf	0x00	big endian byte order
lite_component_count	8	uimsbf	N1	
DSM::ServiceLocation {				
componentId_tag	32	uimsbf	0x49534F46	TAG_ServiceLocation
component_data_length	8	uimsbf	*	
serviceDomain_length	8	uimsbf	0x14	Length of carousel NSAP address
serviceDomain_data()	160	uimsbf	+	table B.26, "DVB Carousel NSAP Address"
CosNaming::Name() {				pathName
nameComponents_count	32	uimsbf	N2	
for (i=0; i<N2; i++) {				
id_length	32	uimsbf	N3	NameComponent id
for (j=0; j<N3 j++) {				
id_data	8	uimsbf	+	
}				
kind_length	32	uimsbf	N4	NameComponent kind
for (j=0; j<N4 j++) {				
kind_data	8	uimsbf	+	as type_id (see table 4-4 in TR 101 202 [i.6])
}				
}				
}				
initialContext_length	32	uimsbf	N5	
for (n=0; n<N5 n++) {				
InitialContext_data_byte	8	uimsbf		
}				
}				
for (n=0;n<N6;n++) {				N6=N1-1
BIOP::LiteComponent{				
componentId_tag	32	uimsbf	+	
component_data_length	8	uimsbf	N7	
for (i=0; i<N7; i++) {				
component_data_byte	8	uimsbf		
}				
}				
}				
}				

Table B.26: DVB Carousel NSAP Address

Syntax	bits	Type	Value	Comment
DVBcarouselNSAPaddress {				
AFI	8	uimsbf	0x00	NSAP for private use
Type	8	uimsbf	0x00	Object carousel NSAP Address.
carouselId	32	uimsbf	+	To resolve this reference a carousel_identifier_descriptor with the same carousel_id as indicated in this field must be present in the PMT signalling for the service identified below.
specifierType	8	uimsbf	0x01	IEEE OUI
specifierData { IEEE OUI }	24	uimsbf	0x00015A	Constant for DVB OUI
dvb_service_location () {				
transport_stream_id	16	uimsbf	+	This may be set to 0x0000 which indicates that the MHP terminal shall not use the transport_stream_id when locating the service. For any other value then this field shall be used.
original_network_id	16	uimsbf	+	
service_id	16	uimsbf	+	(= MPEG-2 program_number)
reserved	32	bslbf	0xFFFFFFFF	
}				
}				

B.2.3.8 BIOP StreamMessage

Table B.27: Restrictions on the BIOP Stream Message

Field	Restrictions	Source
MessageSubHeader::ObjectInfo	The ObjectInfo field contains the DSM::Stream::Info_T structure and optionally other data after the Stream Info structure. MHP terminals may ignore the aDescription_bytes in the DSM::Stream::Info_T structure and the possible other object info data following the structure. Broadcasts may set the duration field to zero to indicate undefined duration.	The present document
MessageSubHeader::ServiceContextList	The MHP terminal may skip the possible serviceContextList structures.	The present document
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 or STR_DVBTIMEL_USE indicating a timebase to be associated with the Stream object. Taps with use STR_NPT_USE shall be interpreted as described in ISO/IEC 13818-6 [16]. Taps with use STR_DVBTIMEL_USE shall be interpreted according to clause B.2.3.10. MHP terminals may ignore possible other Taps (such as BIOP_ES_USE).	The present document

Table B.28: BIOP::StreamMessage syntax

Syntax	bits	Type	Value	Comment
BIOP::StreamMessage() {				
magic	4 × 8	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	1 to 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	+	Flag: 0x00 = false, non-zero = true.
video	8	uimsbf	+	Flag: 0x00 = false, non-zero = true.
data	8	uimsbf	+	Flag: 0x00 = false, non-zero = true.
}				
for (i=0; i<N2-(N3+10); i++) {				
objectInfo_byte	8	uimsbf	+	
}				
serviceContextList_count	8	uimsbf	N4	serviceContextList
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	(note)	see clause B.2.4.4
use	16	uimsbf	+	see clause B.2.3.10 and table 4-12 in DVB Guidelines for Data Broadcasting
assocTag	16	uimsbf	+	
selector_length	8	uimsbf	0x00	no selector
}				
}				

NOTE: If the tap use is STR_NPT_USE then the value of this 16 bit integer corresponds to the value of the contentId field of the NPTReferenceDescriptor that defines the time base for this stream. If the tap use is STR_DVBTIMEL_USE then the value of this 16 bit integer corresponds to the value of the broadcast_timeline_id field of the DVB Timeline that defines the timebase for this stream. For other values of tap use the value of this field is undefined.

B.2.3.9 BIOP StreamEventMessage

Table B.29: Restrictions on the BIOP StreamEvent Message

Field	Restrictions	Source
MessageSubHeader::ObjectInfo	The ObjectInfo field contains the DSM::Stream::Info_T and DSM::Stream::EventList_T structures followed optionally by other object info data (which may be ignored by MHP terminals). See table B.27, "Restrictions on the BIOP Stream Message"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.	The present document
MessageSubHeader::ServiceContextList	The MHP terminal may skip the possible serviceContextList structures.	The present document
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 or STR_DVBTIMEL_USE indicating a timebase to be associated with the StreamEvent object. Taps with use STR_NPT_USE shall be interpreted as described in ISO/IEC 13818-6 [16]. Taps with use STR_DVBTIMEL_USE shall be interpreted according to clause B.2.3.10. There shall be at most one tap with use STR_EVENT_USE STR_STATUS_AND_EVENT_USE or STR_DVBEVENT_USE. This tap indicates the PID where event data relating to the StreamEvent object is broadcast. MHP terminals may ignore possible other taps (such as BIOP_ES_USE).	The present document

Table B.30: BIOP::StreamEventMessage syntax

Syntax	bits	Type	Value	Comment
BIOP::StreamEventMessage() {				
magic	4 × 8	uimsbf	0x42494F50	"BIOP"
biop_version.major	8	uimsbf	0x01	BIOP major version 1
biop_version.minor	8	uimsbf	0x00	BIOP minor version 0
byte_order	8	uimsbf	0x00	big endian byte ordering
message_type	8	uimsbf	0x00	
message_size	32	uimsbf	*	
objectKey_length	8	uimsbf	N1	
for (i=0; i<N1; i++) {				
objectKey_data	8	uimsbf	+	
}				
objectKind_length	32	uimsbf	0x00000004	
objectKind_data	4 × 8	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
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 + 14 + N4 + sum(N5)); 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	(see note)	See clause B.2.4.4
use	16	uimsbf	+	see clause B.2.3.10 and 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	+	
}				
}				

NOTE: If the tap use is STR_NPT_USE then the value of this 16-bit integer corresponds to the value of the contentId field of the NPTReferenceDescriptor that defines the time base for this stream. If the tap use is STR_DVBTIMEL_USE, the value of this 16-bit integer corresponds to the value of the broadcast_timeline_id field of the DVB Timeline that defines the timebase for this stream. If the tap use is STR_DVBEVENT_USE,

Syntax	bits	Type	Value	Comment
the value of this 16-bit integer field corresponds to the value of the synchronized_event_context of all events relevant to this stream. For other values of tap use the value of this field is undefined.				

B.2.3.10 Additional tapUse values

Two additional tapUse values are defined for use in referencing DVB synchronized auxiliary data TS 102 823 [48] from an Object Carousel as shown in table B.31.

Table B.31: tapUse values for referencing DVB synchronized auxiliary data

tapUse	Value
STR_DVBTIMEL_USE	0x8000
STR_DVBEVENT_USE	0x8001

The semantics of the fields of a Tap pointing to a DVB broadcast_timeline_descriptor are described below:

- The use field indicates the use of the Tap. The value of this field shall be STR_DVBTIMEL_USE.
- The value of the id field shall specify the broadcast_timeline_id of the timeline to be referenced.
- The assocTag identifies the connection on which the broadcast_timeline_descriptor is broadcast.
- The selector field shall be empty.

The semantics of the fields of a Tap pointing to a DVB synchronized_event_descriptor are described below:

- The use field indicates the use of the Tap. The value of this field shall be STR_DVBEVENT_USE.
- The value of the id field shall specify the synchronized_event_context of the events to be referenced.
- The assocTag identifies the connection on which the synchronized_event_descriptors are broadcast.
- The selector field shall be empty.

B.2.4 Broadcast timebases and events

MHP terminals are required to support two mechanisms for broadcast timebase delivery: DSM-CC NPT and DVB Timeline.

MHP terminals are also required to support broadcast events as follows:

- DSM-CC scheduled stream events. These are defined with reference to a broadcast timebase defined using either DSM-CC NPT or the DVB Timeline mechanism.
- DSM-CC "do it now" stream events. These are stand-alone events that are independent of a broadcast timebase.
- DVB synchronized events. These are stand-alone events that are independent of a broadcast timebase but which can provide much greater temporal accuracy than DSM-CC "do it now" events.

The BIOP StreamMessage and StreamEventMessage are used within a DSM-CC object carousel to reference timebases and to define events. A BIOP StreamMessage can only be used to reference a broadcast timebase. A BIOP StreamEventMessage can be used to define broadcast events with or without reference to a broadcast timebase.

NOTE: The NPT mechanism and scheduled stream events that depend on it are known to be vulnerable to disruption in many digital TV distribution networks. Existing deployed network equipment that re-generates the STC is unlikely to be aware of NPT and hence will not make the necessary corresponding modification to STC values inside NPT reference descriptors. This may cause stream events scheduled against NPT to fire at the wrong time or to never fire at all. Applications should only use scheduled stream events with NPT when they are confident that the network where they are to be used does not have this problem. DVB Timeline, DSM-CC "do it now" events and events carried within DVB synchronized auxiliary data offer more reliable alternatives to NPT.

B.2.4.1 Stream and StreamEvent messages

B.2.4.1.1 Association with time bases

The id field of the STR_NPT_USE or STR_DVBTIMEL_USE tap of a StreamMessage or StreamEventMessage identifies the timebase associated with that Stream/StreamEvent object. Multiple StreamMessage or StreamEventMessage may be used at the same time to allow subscriptions to multiple timebases of the same service. See clause B.2.4.4, "Broadcast timebases".

B.2.4.1.2 Event names and event ids

In StreamEventMessages the EventList_T defines a sequence of event names that correlates 1:1 to the sequence of event ids in the MessageBody. Within each BIOP::StreamEventMessage the event names uniquely associate to event id values:

- The eventNames_count shall equal eventIds_count.
- The names in the EventList_T are zero-terminated strings.
- The eventID values in the StreamEventMessage correspond to the eventID values carried in StreamEventDescriptors or the synchronized_event_id values in DVB synchronized_event descriptors.

B.2.4.1.3 Stream event life time

In StreamEventMessages the set of events described in the BIOP::StreamEvent message is possibly a subset of the events that may be used by the application during the course of a programme. Therefore, applications may need to accommodate the dynamic change of such messages. Cache transparency (see clause B.5.2.1, "Transparent caching") and version listener mechanisms (see DSMCCObject methods in annex P "(normative): Broadcast Transport Protocol Access") provide applications with the means to do this.

Similarly the set of stream event descriptors being transmitted at any time may not correspond to the set of events described in the BIOP::StreamEventMessage.

The event id for an event name shall not change while the name exists. If a name is removed it shall not be reintroduced within 60 seconds.

B.2.4.2 Stream Descriptors

B.2.4.2.1 NPT Reference descriptor

B.2.4.2.1.1 Syntax

MHP terminals shall interpret this descriptor as it is described in ISO/IEC 13818-6 [16] with the following clarifications and additions.

With regard to contentId:

- This 7 bit field identifies the "timebase" see clause B.2.4.4 "Broadcast timebases".

With regard to scaleNumerator and scaleDenominator:

- 1/1 - means normal play.

  i.e. NPT and STC advance at the same rate.

- 0/m (m > 0) - means the NPT value does not advance.

  As a consequence no scheduled stream events will be raised. However, the "do it now" events continue to be effective.

- 0/0 - means that the scaleNumerator and scaleDenominator fields are not defined in the NPT Reference descriptor and should be derived as described in ISO/IEC 13818-6 [16], clause 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 [16].

With regard to broadcast repetition rate:

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

With regard to postDiscontinuity indicator:

- It is optional for broadcasters to broadcast descriptors with this set to one however if they are broadcast then the following conditions shall be met:

  - It shall be broadcast for at least 5 seconds before the underlying PCR discontinuity.

  - The descriptors with both states of postDiscontinuity indicator shall be carried within the same DSMCC descriptor list section.

  - Within one second of the underlying PCR discontinuity, the NPTReferenceDescriptor must be updated to reflect the new underlying PCR. See `org.dvb.dsmcc.NPTDiscontinuityEvent` in annex P "(normative): Broadcast Transport Protocol Access".

- It is optional for MHP terminals to take advantage of this signalling but terminals which do not take advantage of this must ignore descriptors with postDiscontinuity indicator set to 1.

B.2.4.2.2 Stream event descriptor

B.2.4.2.2.1 Association of event ids to event time

Where the timebase for a scheduled stream event is provided by NPT, the eventNPT field conveys the NPT value at which the event will occur (or has occurred). Where the timebase for an event is provided by DVB Timeline, the eventNPT field conveys the tick value at which the event will occur (or has occurred) and shall be interpreted in units of the tick_rate used to define the DVB Timeline.

Each StreamEventDescriptor provides a single association between an eventID and an eventNPT. If the MHP terminal detects a change in the eventNPT associated with a value of eventID this redefines the time at which the event should fire.

MHP terminals shall ignore scheduled events where the eventNPT has passed.

See also clauses B.2.4.2.4.1 "number range for NPT" and B.2.4.2.2.3 "Signalling of "do it now events"".

B.2.4.2.2.2 Re-use of event ids

Event ID values may be re-used any number of times. For example, after an event has fired then stream event descriptors with the same eventID but different eventNPT may be broadcast.

B.2.4.2.2.3 Signalling of "do it now events"

ISO/IEC 13818-6 [16] 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 clause B.2.4.3.5, "Encoding of table id extension").

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

B.2.4.2.2.4 Private data

The contents of the privateDataByte field do not need to be interpreted by the MHP terminal. However, the application can access the privateDataByte field using the `org.dvb.dsmcc.StreamEvent.getEventData` method.

B.2.4.2.3 Unused descriptors

MHP terminals may ignore the following descriptors if present:

- NPT Endpoint descriptor.
- Stream Mode descriptor.

B.2.4.2.4 Clarification of number encoding

B.2.4.2.4.1 number range for NPT

There is some ambiguity in ISO/IEC 13818-6 [16] regarding the data type used to carry NPT values in the signalling (tcimsbf or uimsbf). The following requirements insulate this profile from this ambiguity:

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

B.2.4.2.4.2 number range for scaleDenominator

There is some ambiguity in ISO/IEC 13818-6 [16] regarding the data type used to carry `scaleDenominator` values in the signalling (tcimsbf or uimsbf). The following requirements insulate this profile from this ambiguity:

- The range of values used shall be in the range 0 to 0x7FFF (which is unambiguous for both tcimsbf or uimsbf).

B.2.4.3 DSM-CC Sections carrying Stream Descriptors

B.2.4.3.1 Section version number

The section version number field increments to reflect changes in stream descriptor(s) carried by sections with the same value of table_id (0x3D) and table_id_extension.

The version number shall increment for reasons including the change in value of eventNPT for a given eventId.

B.2.4.3.2 Single firing of "do it now" events

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

B.2.4.3.3 Section number

For the present document MHP terminals shall only consider section number zero.

B.2.4.3.4 DSM-CC sections for DSMCC_descriptor_list()

If the `table_id` field equals 0x3D the `current_next_indicator` bit shall be set to "1".

B.2.4.3.5 Encoding of table id extension

The section's table id extension field provides information on the stream descriptor(s) carried by the section.

Table B.32: 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 to 0]	
0	0	eventID[13 to 0]	Section carries a single "do it now" event
0	1	xx xxxx xxxx xxxx	Section carries NPT reference descriptors
1	0	xx xxxx 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 the present document). - NPT endpoint descriptor (can be ignored in the present document).
1	1	reserved for future use	

The value of eventID for "do it now" events shall be in the range 0x0001 to 0x3FFF. The value of eventID for scheduled events shall be in the range 0x8000 to 0xBFFF. The value 0 is not allowed (see clause 5.5.2.2.1 in ISO/IEC 13818-6 [16]).

B.2.4.4 Broadcast timebases

Multiple concurrent timebases may be defined for a single MPEG program but only a single time base is allowed to progress at any instant (the other timebases shall be paused). Timebases can be provided by two means: DSM-CC NPT and DVB Timeline.

B.2.4.4.1 DSM-CC NPT

The relationship between each NPT timeline and the MPEG timebase (STC) is defined by an NPTReferenceDescriptor. The contentId field of the NPTReferenceDescriptor (a 7-bit unsigned integer) identifies the timebase.

The value of the id field of the STR_NPT_USE tap (a 16 bit unsigned integer) of a StreamMessage or StreamEventMessage identifies the timebase associated with that Stream/StreamEvent object. Multiple StreamMessage or StreamEventMessage may be used at the same time to allow subscriptions to multiple timebases of the same service.

In this profile NPTReferenceDescriptors can indicate two states:

- Non-paused: The scaleNumerator and scaleDenominator are both non-zero.
- Paused: The scaleNumerator is zero and the scaleDenominator is non-zero.

When a timebase is signalled as paused then the NPT value for that timebase is frozen at NPT_Reference (as specified by equation 8-4 in DSM-CC).

All of the NPTReferenceDescriptors for all of the currently signalled timebases shall be carried in a single DSMCC_descriptor_list section and shall be transmitted at least once every second. In any such set of NPTReferenceDescriptors at most one shall be non-paused and there shall be at most one instance of each value of contentId. The set of signalled timebases can change through time. When a timebase is not signalled then the behaviour of the MHP terminal shall be identical to that of the timebase being paused.

Timebases can be added or subtracted from the current set. The set of current timebases can be empty.

A value of contentId shall not be reused for a new timebase within 60 seconds of the removal of the timebase.

In normal use the NPT of a non-paused timebase progresses at a constant rate. Discontinuities should either be the results of errors in the broadcast or transient conditions (for example, while an NPT reference generator catches up with an MPEG PCR discontinuity). Transient discontinuities should be tolerated by the MHP Terminal. The behaviour of the MHP terminal when subject to a permanent discontinuity is not specified, apart from the generation of the NPTDiscontinuityEvent to registered listeners.

The broadcaster shall start generating corrected NPTReferenceDescriptors within at most 1 second of a PCR discontinuity (ideally the descriptors should be generated before the PCR discontinuity). If the receiver is sampling the NPTReferenceDescriptor at the lowest allowed rate (once every 5 seconds) then the receiver may not receive a correct NPTReferenceDescriptor for 5 seconds. During this period the receiver should linearly extrapolate the NPT from previous NPT values in the expectation that a corrected NPTReferenceDescriptor will be delivered shortly. See also clause B.2.4.2.1.1, "Syntax" on use of the postDiscontinuity.

There is a window of uncertainty around a segment of paused timebase due to the time taken for all receivers to acquire the new NPTReferenceDescriptor. During this window scheduled events cannot be used reliably.

NOTE: It is suggested that broadcasters use "do it now" events near junctions between different timebases.

B.2.4.4.2 DVB Timeline

The relationship between each DVB Timeline and the MPEG timebase (STC) is defined using the broadcast_timeline_descriptor carried in DVB Synchronized Auxiliary Data TS 102 823 [48]. The broadcast_timeline_id field of the broadcast_timeline_descriptor (an 8-bit unsigned integer) identifies the timebase.

The value of the id field of the STR_DVBTIMEL_USE tap (a 16-bit unsigned integer) of a StreamMessage or StreamEventMessage identifies the timebase associated with that Stream/StreamEvent object. Multiple StreamMessages or StreamEventMessages may be used at the same time to allow subscriptions to multiple timebases of the same service.

In this profile, broadcast_timeline_descriptors can indicate two states:

- Non-paused: The running_status field set to "Running".
- Paused: The running_status field set to "Paused".

The DVB Timeline referenced may be defined using either the direct or offset encoding method specified in TS 102 823 [48].

Descriptors other than the broadcast_timeline_descriptor may be present within the Synchronized Auxiliary Data stream. MHP terminals shall skip descriptors which they do not support and continue processing the next descriptor. Not supported descriptors include those defined but not supported (e.g. the TVA_id descriptor in devices not required to support that descriptor by another specification) and descriptors whose tag value is either reserved or user private. See also clause B.2.4.5.3.

The Synchronized Auxiliary Data specification TS 102 823 [48] describes the management of DVB Timeline discontinuities.

MHP terminals shall comply with the recommendations for receivers in clause D.5 of TS 102 823 [48] including those defined by "should" in that clause. MHP terminals shall not fire events synchronized to a DVB Timeline in circumstances where the state of the timeline is unknown or where the timeline has been removed from the PMT of the service in which it is carried.

B.2.4.5 Broadcast events

B.2.4.5.1 DSM-CC "do it now" stream events

MHP terminals shall support DSM-CC "do it now" stream events as defined in clause B.2.4.2.2. This provides a means for delivering stand-alone events without the need for a broadcast timebase.

As these events are delivered in the payload of an MPEG Section, they cannot be accurately synchronized with linear media streams.

B.2.4.5.2 DSM-CC scheduled stream events

MHP terminals shall support DSM-CC scheduled stream events as defined in clause B.2.4.2.2. These are defined with reference to a broadcast timebase defined using either DSM-CC NPT or the DVB Timeline mechanism.

For DSM-CC scheduled stream events, the following usage scenarios are envisaged:

- A single continuous timebase (i.e. a single progressing value of time) can be used. In this case, the broadcast is logically a single continuing interactive production, and the broadcaster is responsible for pre-processing the applications, etc. before broadcast to ensure that they are suitable.
- The signal received by the MHP terminal can include a unique timebase for each programme needing one. This timebase could be suspended during any insertion into a programme and discontinued at the end of the programme.

B.2.4.5.3 DVB synchronized events

MHP terminals shall support DVB synchronized events as defined by TS 102 823 [48]. This mechanism allows for the accurate generation of events without the need for a timebase.

DVB synchronized events are defined using the synchronized_event_descriptor.

The synchronized_event_context field of this descriptor identifies the application-specific context for a set of events and is referenced using the id field of the STR_DVBEVENT_USE tap of a DSM-CC StreamEventMessage.

The contents of the synchronized_event_data_byte field does not need to be interpreted by the MHP terminal. However, the application can access the synchronized_event_data_byte field using the org.dvb.dsmcc.StreamEvent.getEventData method.

MHP terminals shall support the cancellation of DVB synchronized events that have not yet fired using the synchronized_event_cancel_descriptor.

Cancellation shall be supported for individual events and for sets of events with a common synchronized_event_context.

Descriptors other than the synchronized_event_descriptor and synchronized_event_cancel_descriptor may be present within the Synchronized Auxiliary Data stream. MHP terminals shall skip descriptors which they do not support and continue processing the next descriptor. Not supported descriptors include those defined but not supported (e.g. the TVA_id descriptor in devices not required to support that descriptor by another specification) and descriptors whose tag value is either reserved or user private. See also clause B.2.4.4.2.

DVB synchronized events have a specified presentation time, determined by the presentation timestamps of the auxiliary data structures carrying synchronized_event_descriptors and the reference_offset_ticks field of the synchronized_event_descriptors. As such, the events are synchronized to a point in time within linear media streams such as video or audio. Receivers shall deliver events to applications timed so as to retain this synchronization with the linear media.

B.2.4.6 Monitoring broadcast timebases and events

B.2.4.6.1 Timebase reference monitoring

When applications have registered for timebase stimulated events, the MHP terminal shall allocate resources sufficient to ensure that updates to the set of timebases is detected within 5 seconds for conformant broadcasts.

B.2.4.6.2 Timebase stimulated event monitoring

When applications have registered for timebase stimulated events the MHP terminal shall allocate resources sufficient to ensure that updates to the set of timebase stimulated events is detected within 5 seconds for conformant broadcasts. So, if an event is introduced or the time at which it is specified to fire is changed then the MHP terminal will respect this change within 5 seconds. If the fire time for an event changes less than 5 seconds before it was previously scheduled to fire then there is no guarantee that all receivers will detect the change in time.

The receiver shall deactivate any event listeners dependant on a timebase (and may free resources associated with those listeners) if:

- the timebase is an NPT timebase and it is deleted (reference to it is removed from the set of `NPTReferenceDescriptors`);
- the timebase is an NPT timebase and a discontinuity is detected (i.e. `NPTDiscontinuityEvent` generated) in that timebase;
- a service selection operation changes the current service, either through the MHP API or through the service selection feature of the MHP navigator.

B.2.4.6.3 DSM-CC "do it now" stream events

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

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

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

B.2.4.6.4 DSM-CC scheduled stream events

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

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

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

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

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

B.2.4.6.5 number of timebase components

The MHP terminal is only required to monitor a single timebase component. So, if events from more than one DSM-CC StreamEventMessages are subscribed, each StreamEventMessage that references a timebase shall reference the same type (NPT or DVB Timeline) and shall contain a STR_NPT_USE or STR_DVBTIMEL_USE tap specifying the same association tag.

B.2.4.6.6 DVB synchronized events

DVB synchronized events are transient events (i.e. only broadcast for a short period of time). Accordingly, MHP terminals need to make special efforts to ensure a high probability that they can be reliably received.

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

MHP terminals shall dedicate a filter to monitoring the possible transmission of such events while there are any applications subscribed to them.

MHP terminals are not required to monitor DSM-CC "do it now" stream events at the same time as monitoring DVB synchronized events.

B.2.5 Assignment and use of transactionId values

B.2.5.1 Informative Background

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

NOTE: The term "incremented" is used in the DSM-CC specification. Within the scope of the present document 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 the present document a set of DownloadInfoIndication message transmitted on a single PID and the DownloadDataBlock messages carrying the modules described in the DownloadInfoIndication messages. The DownloadDataBlock messages may be spread on other elementary streams than the DownloadInfoIndication messages. The DownloadServerInitiate message in the context of object carousels is considered to be part of the top level of the object carousel and not associated with any data carousel.

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

B.2.5.2 DVB semantics of the transactionId field

The transactionId has been split up into a number of sub-fields defined in table B.33. 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.33: 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 to 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 to 29	User-defined	Version	This shall be incremented every time the control message is updated. The value by which it is incremented should be one.
30 to 31	Bit 30 - zero Bit 31 - non-zero	Originator	This is defined in the DSM-CC specification ISO/IEC 13818-6 [16] 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 to 15) shall be matched.**

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

B.2.6 Mapping of objects to data carousel modules

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

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

- The size of a module that contains multiple objects should not exceed 65 536 bytes when decompressed (i.e. when the file has been decompressed from the file transport but before the content decoding has started). MHP terminals complying to the present document are only required to handle modules containing multiple objects where the module size when decompressed is 65 536 bytes or less. Modules containing a single file message can exceed 65 536 bytes with upper size only limited by the memory resources in the MHP terminal.

- In addition to the limitations imposed by the 65 536 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.

Table B.34 shows the syntax of the compressed_module_descriptor.

Table B.34: 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 [28].

The MHP terminal shall support the Deflate compression algorithm as specified in RFC 1951 [29]. This is signalled by setting the least significant nibble of the compression_method to 0x8 (i.e. compression_method is xxxx1000). The MHP terminal is not required to support other compression algorithms.

B.2.8 Mounting an Object Carousel

The ServiceGateway object is the root directory of the file system delivered by an Object Carousel and must be acquired before any other object can be downloaded. This may be achieved by two compatible mechanisms. The signalling of which mechanisms are being supported by a broadcast is provided by the carousel_identifier_descriptor.

In the present document the use of the carousel_identifier_descriptor for signalling is mandatory in the second descriptor loop of a PMT (corresponding to a PID on which the DSI message for an Object Carousel is broadcast, i.e. the boot-PID). The consequence is that if a PMT second descriptor loop contains a data_broadcast_id_descriptor that provides signalling for the present document, it shall also contain a carousel_identifier_descriptor.

NOTE: A single PID only contains messages from a single Object Carousel and so only one carousel_identifier_descriptor is present in any second descriptor loop. However, a single service may contain more than one Object Carousel. Consequently, the carousel_identifier_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_identifier_descriptor and shall be sufficient for all MHP terminals.

See also clause 10.2, "Program Specific Information".

A broadcast may also contain additional information in the carousel_identifier_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_identifier_descriptor

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

Table B.35: Carousel identifier descriptor syntax

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

carousel_id: This 32 bit field identifies the object carousel with the corresponding carouselId, with the carousel_identifier_descriptor carrying the broadcaster's OrgID in the most significant 24 msbs if carousel is sharable across multiple transport streams or between 0 - 255 otherwise.

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 [28] used to compress the module. It is equivalent to compression_method carried in the compressed_module_descriptor in the DII.

OriginalSize: This 32 bit integer is the size of the data (in bytes) carried by the module before it was compressed. It is equivalent to original_size carried in the compressed_module_descriptor in the DII.

If the module has not been compressed the values of OriginalSize and ModuleSize shall be equal and the value of CompressionMethod is not defined.

TimeOut: This 8 bit integer specifies the timeout in seconds for acquisition of all blocks of the module.

ObjectKeyLength: This 8 bit integer specifies the number of bytes of ObjectKeyData.

ObjectKeyData: These 8 bit values form an octet string that identifies the BIOP message that is the ServiceGateway message.

B.2.8.2 DVB-J mounting of an object carousel

DVB-J causes an object carousel to be mounted using `ServiceDomain.attach()`. It can be unmounted using `ServiceDomain.detach()`.

An application manager is also allowed to call these methods implicitly when launching or killing an application in order to access the signalled base directory of the application.

B.2.9 Unavailability of a carousel

Broadcast carousels become permanently unavailable due to changes in the signalling including the following:

- The component signalled as carrying the DSI is removed from the PMT.
- The value of carousel ID associated with the carousel changes.
- The program disappears from the PAT.
- After an implementation dependent time general failure of the signalling (e.g. non-transmission of the PMT).

Additionally, carousels also become permanently unavailable when loss of connection to a temporarily disconnected carousel becomes permanent, as defined in clause 6.2.5.3.

B.2.10 Delivery of Carousel within multiple services

Carousels shall be considered identical if, in the PMTs of the services, all the following hold:

Either:

a) Both services are delivered within the same transport stream, and:

- Both services list the boot component of the carousel on the same PID.
- The carousel_identifier_descriptor for the carousel are identical in both services (so the carousels have the same carousel_Id and boot parameters).
- All association tags used in the carousel map to the same PIDs in both services.

In this case, the carousel is transmitted over a single path, but the services are allowed to reference the carousel via a number of routes, including deferral to a second PMT via deferred association tags.

Or:

b) Both services are delivered over multiple transport streams, and:

- The carousel_id in the carousel_identifier_descriptor is in the range of 0x100 - 0xffffffff (containing the broadcaster's OrgID in the most significant 24 msbs of carousel_id).
- The carousel_identifier_descriptor for the carousel are identical in both services (so the carousels have the same carousel Id and boot parameters).

B.3 AssociationTag Mapping

B.3.1 Decision algorithm for association tag mapping

B.3.1.1 TapUse is **not** BIOP_PROGRAM_USE

Figure B.1 illustrates the decision tree for identifying the elementary stream(s) by which the object carousel is distributed.

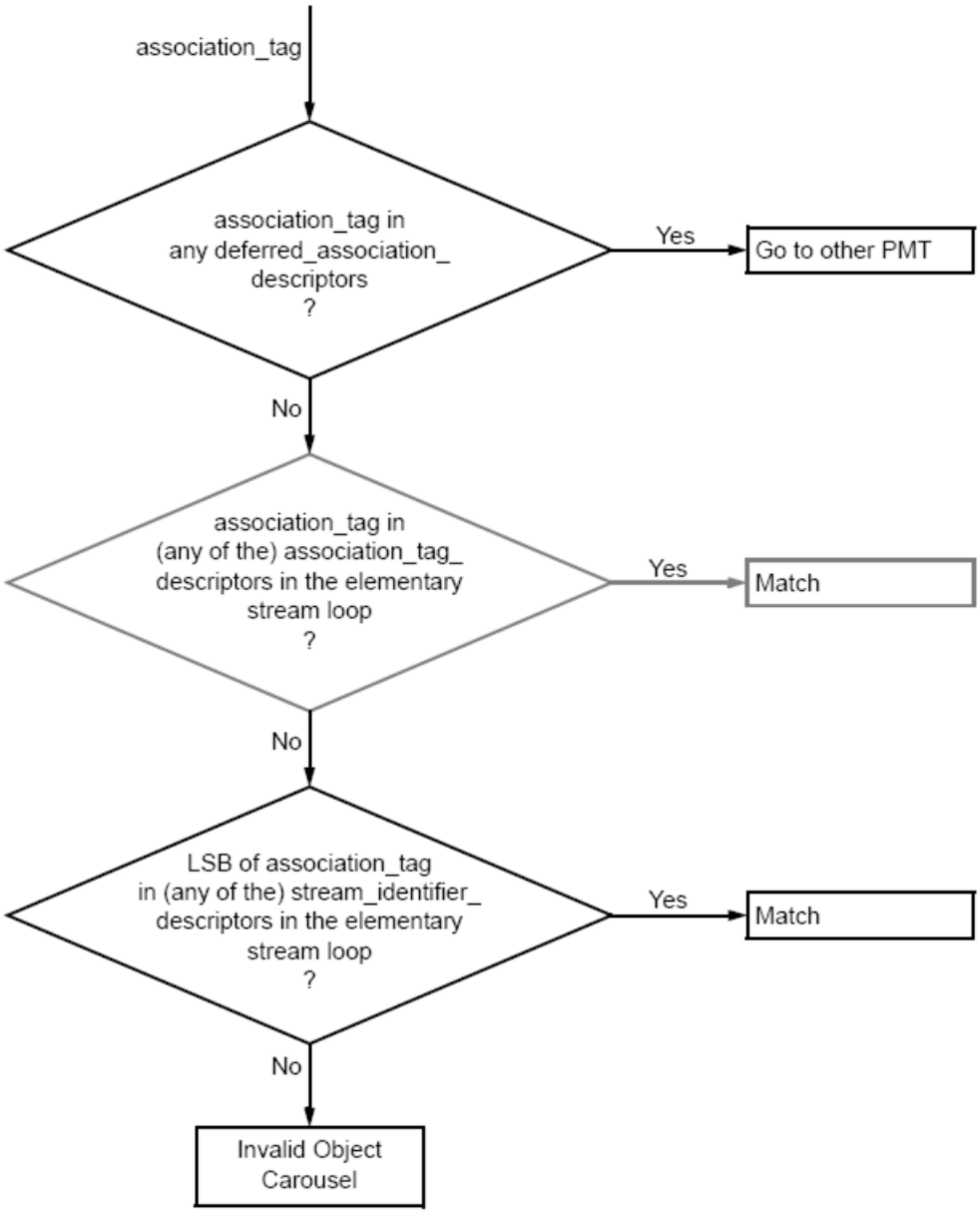

Figure B.1: Object Carousel ES identification decision tree

In the present document, 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 (as defined in EN 300 468 [4]) shall still be present and the tag values shall be set consistently in each descriptor. This restriction simplifies the decision tree above so that the second decision can be skipped.

B.3.1.2 TapUse **is** BIOP_PROGRAM_USE

The decision tree in clause B.1 is not followed when resolving a `BIOP_PROGRAM_USE` tap as the only valid broadcast encoding is for a tap of use `BIOP_PROGRAM_USE` to resolve to `deferred_association_tags_descriptor` in the PMT even if the `deferred_association_tags_descriptor` identify the current service (i.e `stream` or `streamEvent` reference itself). If this resolution fails then the service from which the object carousel is mounted shall be returned as the referenced service.

B.3.2 DSM-CC association_tags to DVB component_tags

The component_tag in a PMT's stream_identifier_descriptor (as defined in EN 300 468 [4]) 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 [16]) 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_tags_descriptor required by the present document is the adaptation of the ISO/IEC 13818-6 [16] descriptor defined in TR 101 202 [i.6]. This latter definition standardizes a mechanism to signal the original network id.

When attempting to map an assocation_tag to an elementary stream the assocation_tag must first be checked against any deferred_association_tags_descriptors in the current PMT (current in this context means the PMT of the service within which the assocation_tag is being mapped). If the association_tag matches any of the association_tags present in a deferred_assocation_tags_descriptor then the matching process proceeds to the service indicated in that descriptor. The MHP terminal is not required to continue its search beyond this second service.

If the transport_stream_id field in the deferred_association_tags_descriptor is set to 0x0000 then it shall be ignored and the MHP terminal is free to choose which transport stream ID it selects when obtaining a service.

B.3.3 deferred_association_tags_descriptor

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

B.4 Example of an Object Carousel (informative)

Figure B.2 illustrates an object carousel that is distributed over three elementary streams belonging to the same service.

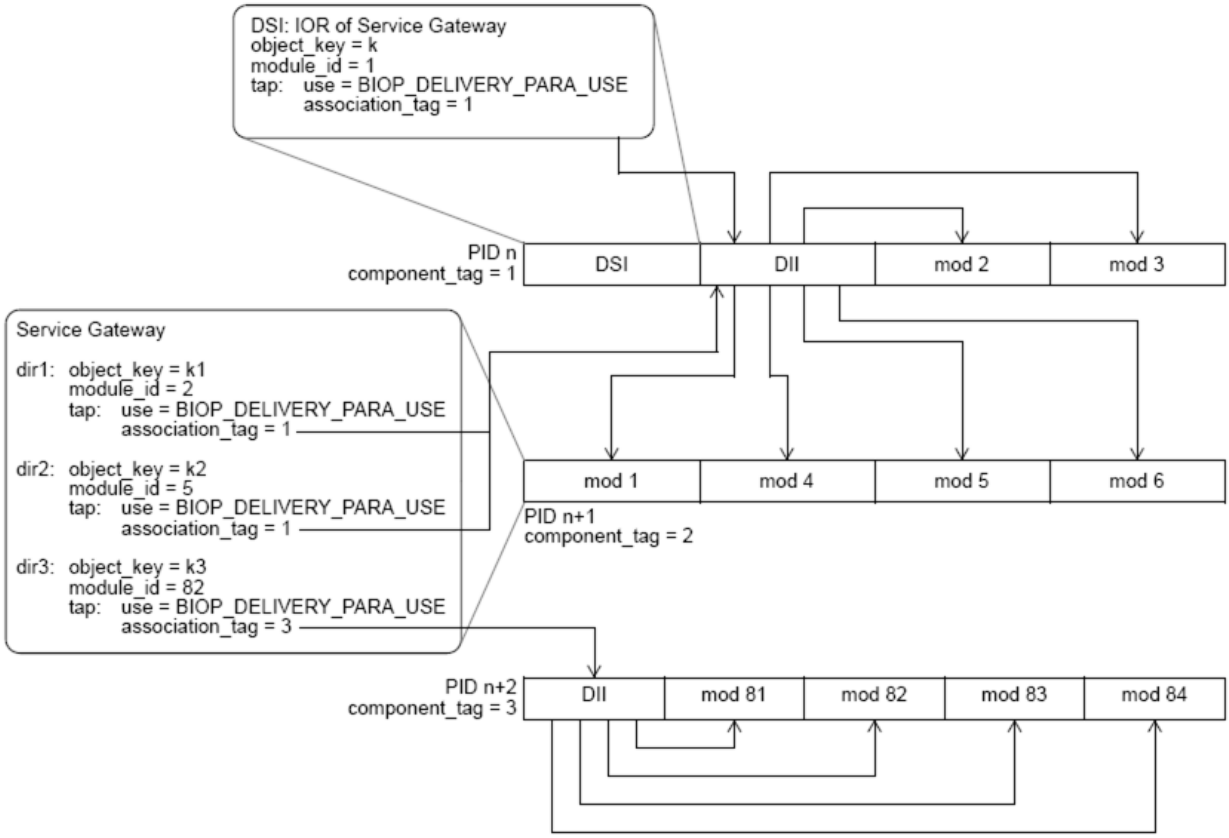

Figure B.2: Example carousel

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.

NOTE: 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

GEM [1], clause B.5 is included in the present document.

Annex C (informative): Bibliography

GEM [1], annex C is included in the present document.

Annex D (normative): Text presentation

GEM [1], annex D is included in the present document, with the notes and modifications detailed in the following clauses.

D.1 Font Technology

GEM [1], clause D.2.2.1 references clause 7.4. For the present document this reference is to be interpreted as referring to clause 7.4, "Downloadable fonts" in the present document.

Annex E (normative): Character set

E.1 Basic Euro Latin character set

GEM [1], annex E is included in the present document.

Annex F (informative): Authoring and Implementation Guidelines

GEM [1], annex F is included in the present document with the addition of the following clause.

F.4 Authoring guidelines for DVB HTML

F.4.1 CSS2 Authoring guidelines

The selection of the CSS2 subset in GEM has been influenced by the requirements of the conformance clauses as defined by the W3C. As such, some style properties and selectors have been selected in order to respect this requirement, although they might not be fully adapted to the TV environment and its constraints. The objective of this section is to provide content authors with some guidelines on the potential impacts the use of some selectors or properties might have. Some of those guidelines are due to CPU or other limitations that may with generations of receivers at the time of the writing. This might evolve with the arrival of new generations, however content authors should be aware of the potential existence of different generations of receivers in the field.

F.4.1.1 Selectors

Content authors are advised that the use of contextual selectors - descendent, child, adjacent sibling, :first-child - as well as the attribute and the attribute value selector might cause the pattern matching process to be slow when used through the DOM APIs. For better matching efficiency, the type, id and class selectors should be favoured.

Content authors are advised that the use of the :first-letter and :first-line pseudo-elements might cause the rendering to be slow. See also guidelines for content generation and the :before and :after selectors.

F.4.1.2 Properties

F.4.1.2.1 Generalities

On a general basis, use of the "auto" value for properties should be avoided and explicit values should be preferred wherever possible.

F.4.1.2.2 Visual Formatting Model

F.4.1.2.2.1 "display"

Use of the "table-header-group", "table-row-group", "table-footer-group", "table-column" and "table-column-group" values of this property might cause the rendering to be slow. These values are meant to make elements behave like the table elements. Their functionality should be well understood before using them.

F.4.1.2.2.2 "float", "clear"

Use of those properties might slow down the positioning operation. Also, for better content predictability and interoperability, absolute positioning should be favoured (see below).

F.4.1.2.2.3 "position"

Use of the "fixed" value of this property might cause the associated rendering to be slow, especially in the case where boxes with different stack levels and with some level of translucency are used through the "z-index" property. On a general basis, for better content predictability and interoperability, absolute positioning should be favoured. Also, Content authors should not make any assumptions on the way the fixed positioning functionality interacts with scrolling.

F.4.1.2.2.4 Generated Content, automatic numbering, and lists

On a general level, content generation provides convenient functionality to the content author. However, this is at the cost of increased complexity in the receiver implementation. Content authors are then advised that the use of the associated selectors (:before and :after) and properties ("content", "counter-increment", "counter-reset", "quotes") might slow down the overall rendering.

F.4.1.2.2.5 "marker-offset", "display" (value "marker") :

Use of the list properties such as "list-style-type", "list-style-image", "list-style-position", and "list-style" should be favoured to markers because of the induced complexity of the latter, its relationships with content generation and consequently its potential slowness in the rendering.

F.4.1.2.3 Colours and Background

F.4.1.2.3.1 "background-attachment"

Use of the "fixed" value of this property might cause the associated rendering to be slow. Content authors should not make any assumptions on the way the fixed positioning functionality interacts with scrolling.

Moreover, content authors should not make assumption on the presence of this functionality in a receiver since it is optional in CSS2.

F.4.1.2.4 Visual Effects

F.4.1.2.4.1 "overflow"

Use of the "scroll" or "auto" values of this property might not cause the presence of scrolling mechanisms as used in the desktop environment. Due to ergonomic reasons, implementations might choose other mechanisms.

F.4.1.2.5 Fonts

F.4.1.2.5.1 "font"

Content authors should not make any assumption on the expected results when using the "caption", "icon", "menu", "message-box", "small-caption", "status-bar" values of the "font" property, which have no associated semantics in a DVB-GEM environment.

F.4.1.2.6 Text

F.4.1.2.6.1 "text-align"

Use of the "justify" value of this property might slow down the rendering because of the induced complexity. Depending on the algorithm used by the receiver, use of this value might also affect interoperability. Content authors should be aware that conformant receivers might interpret this value as being "left" or "right" depending on the directionality of the language.

Use of a string value for this property in a table cell might also slow down the rendering and in general might be unpredictable, for example when the content flows on more than one line.

F.4.1.2.6.2 "text-shadow"

Use of the optional blur radius after the shadow offset might slow down the rendering because of the induced complexity and may not be visible on a low resolution screen. Content authors should be advised that interoperability might be affected since the algorithm for computing the blur effect has not been specified.

F.4.1.2.7 User Interface

F.4.1.2.7.1 User preferences for colours

Content authors should not make any assumption on the expected results when using the system colours which have no associated semantics in a DVB-GEM environment. Those colours are: ActiveBorder, ActiveCaption, AppWorkspace, Background, ButtonFace, ButtonHighlight, ButtonShadow, ButtonText, CaptionText, GrayText, Highlight, HighlightText, InactiveBorder, InactiveCaption, InactiveCaptionText, InfoBackground, InfoText, Menu, MenuText, Scrollbar, ThreeDDarkShadow, ThreeDFace, ThreeDHighlight, ThreeDLightShadow, ThreeDShadow, Window, WindowFrame, WindowText.

Annex G (normative): Minimum Platform Capabilities

G.1 Graphics

GEM[1], clause G.1 is included in the present document, with the notes and modifications detailed in the following clauses.

G.1.1 Device resolution for Standard Definition

For MHP, the bullet points of GEM [1], clause G.1.1 that discuss refer to "the appropriate resolution with the terminal in standard definition mode" shall mean "either 720 × 576 with 64/45 pixel aspect ratio (for 625-line / 50Hz markets) or 720 × 480 with 32/27 pixel aspect ratio (for 525-line / 60Hz markets)".

G.1.3 Minimum Colour Lookup Table

NOTE: Previous versions of MHP defined minimum color capabilities based on an indexed color model with a precise definition of a color lookup table (CLUT). MHP now requires a "true colour" color representation with at least 4 bits per pixel for each of R. G and B.

G.2 Audio

GEM [1], clause G.2 is included in the present document.

G.3 Video

GEM [1], clause G.3 is included in the present document.

G.4 Resident fonts and text rendering

GEM [1], clause G.4 is included in the present document.

G.5 Input events

GEM [1], clause G.5 is included in the present document with the following notes and modifications:

- Input events `VK_TELETEXT` and `VK_COLORED_KEY_3`, labelled "optional" in GEM [1], are required in MHP.

G.6 Memory

GEM [1], clause G.6 is included in the present document, with the following notes and modifications.

The bullet point requiring enough memory to load a PNG image whose size is the same as the resolution of an `HGraphicsDevice` at standard definition mode, as discussed in clause G.1 is replaced with the following:

- Enough memory to successfully load and display any full screen (at standard definition graphics resolution) 8-bit PNG image (conforming to clause 15.1, "PNG - restrictions") from a file which contains just the mandatory information and excludes any optional extension fields or chunks.

G.7 Other resources

GEM [1], clause G.7 is included in the present document.

Annex H (normative): Extensions

GEM [1], annex H is included in the present document.

Annex I (normative): DVB-J fundamental classes

GEM [1], annex I is included in the present document.

Annex J (normative): DVB-J event API

GEM [1], annex J is included in the present document.

Annex K (normative): DVB-J persistent storage API

GEM [1], annex K is included in the present document.

Annex L (normative): User Settings and Preferences API

GEM [1], annex L is included in the present document.

Annex M (normative): SI Access API

M.1 Unicode

References to "Unicode" in the following API description shall be interpreted as references to ISO 10646-1 [12].

Package

org.dvb.si

Description

Provides access to DVB service information.

General Design

Many of the methods in this package use a common design template. Asynchronous method calls to retrieve data from the network all return an instance of the `SIRequest` class. Applications may use this to inquire about or terminate the retrieval operation. When the retrieval operation completes, an instance of a subclass of `SIRetrievalEvent` will be sent to the `SIRetrievalListener` which the application passed in as a parameter to the original method call to retrieve data from the network. When constructing these events, the platform shall provide as the request parameter to the event constructor, the same instance of the `SIRequest` class as was returned by the original method call which started the retrieval operation. This `SIRequest` instance shall be returned by the `getSource` method on such events.

Class Summary	
Interfaces	
`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, SITransport-Stream, 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.

Class Summary	
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.
TextualServiceIdentifierQuery	An interface that can be implemented by objects representing DVB services.
Classes	
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. due to the necessary resources being all taken up by the calling application or other applications.
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. numeric identifiers out of range, etc.)
SIInvalidPeriodException	This exception is thrown when a specified period is invalid (for example, start time is after the end time).

org.dvb.si

Descriptor

Declaration

```
public class Descriptor
java.lang.Object
  |
  +--org.dvb.si.Descriptor
```

Description

This class represents a descriptor within a sub-table.

A descriptor consist of three fields: a tag, a contentLength and the content.

The tag uniquely identifies the descriptor type. The content length indicates the number of bytes in the content. The content consists of an array of bytes of length content length. The data represented by the content is descriptor type dependent.

See Also:

`DescriptorTag`

Constructors

Descriptor()

```
protected Descriptor()
```

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

Methods

getByteAt(int)

```
public byte getByteAt(int index)
        throws IndexOutOfBoundsException
```

Get a particular byte within the descriptor content.

Parameters:

`index` - index to the descriptor content. Value 0 corresponds to the first byte after the length field.

Returns:

The required byte.

Throws:

`java.lang.IndexOutOfBoundsException` - if index < 0 or index >= ContentLength.

getContent()

```
public byte[] getContent()
```

Get a copy of the content of this descriptor (everything after the length field).

Returns:

a copy of the content of the descriptor.

getContentLength()

```
public short getContentLength()
```

This method returns the length of the descriptor content as coded in the length field of this descriptor.

Returns:

The length of the descriptor content.

getTag()

```
public short getTag()
```

Get the descriptor tag. The value returned shall be the actual value used and is not limited to the values defined in `DescriptorTag`.

Returns:

The descriptor tag (the most common values are defined in the DescriptorTag interface).

See Also:

`DescriptorTag`

org.dvb.si

DescriptorTag

Declaration

`public interface` **`DescriptorTag`**

Description

This interface defines constants corresponding to the most common descriptor tags.

See Also:

`Descriptor`

Fields

BOUQUET_NAME

`public static final short` **`BOUQUET_NAME`**

Constant value for the descriptor tag as specified in EN 300 468 [4].

CA_IDENTIFIER

`public static final short` **`CA_IDENTIFIER`**

Constant value for the descriptor tag as specified in EN 300 468 [4].

CABLE_DELIVERY_SYSTEM

`public static final short` **`CABLE_DELIVERY_SYSTEM`**

Constant value for the descriptor tag as specified in EN 300 468 [4].

COMPONENT

`public static final short` **`COMPONENT`**

Constant value for the descriptor tag as specified in EN 300 468 [4].

CONTENT

`public static final short` **`CONTENT`**

Constant value for the descriptor tag as specified in EN 300 468 [4].

COUNTRY_AVAILABILITY

`public static final short` **`COUNTRY_AVAILABILITY`**

Constant value for the descriptor tag as specified in EN 300 468 [4].

DATA_BROADCAST

`public static final short` **`DATA_BROADCAST`**

Constant value for the descriptor tag as specified in EN 300 468 [4].

EXTENDED_EVENT

`public static final short` **`EXTENDED_EVENT`**

Constant value for the descriptor tag as specified in EN 300 468 [4].

FREQUENCY_LIST

```
public static final short FREQUENCY_LIST
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

LINKAGE

```
public static final short LINKAGE
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

LOCAL_TIME_OFFSET

```
public static final short LOCAL_TIME_OFFSET
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

MOSAIC

```
public static final short MOSAIC
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

MULTILINGUAL_BOUQUET_NAME

```
public static final short MULTILINGUAL_BOUQUET_NAME
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

MULTILINGUAL_COMPONENT

```
public static final short MULTILINGUAL_COMPONENT
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

MULTILINGUAL_NETWORK_NAME

```
public static final short MULTILINGUAL_NETWORK_NAME
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

MULTILINGUAL_SERVICE_NAME

```
public static final short MULTILINGUAL_SERVICE_NAME
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

NETWORK_NAME

```
public static final short NETWORK_NAME
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

NVOD_REFERENCE

```
public static final short NVOD_REFERENCE
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

PARENTAL_RATING

```
public static final short PARENTAL_RATING
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

PARTIAL_TRANSPORT_STREAM

```
public static final short PARTIAL_TRANSPORT_STREAM
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

PRIVATE_DATA_SPECIFIER

```
public static final short PRIVATE_DATA_SPECIFIER
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

SATELLITE_DELIVERY_SYSTEM

```
public static final short SATELLITE_DELIVERY_SYSTEM
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

SERVICE

```
public static final short SERVICE
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

SERVICE_LIST

```
public static final short SERVICE_LIST
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

SERVICE_MOVE

```
public static final short SERVICE_MOVE
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

SHORT_EVENT

```
public static final short SHORT_EVENT
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

SHORT_SMOOTHING_BUFFER

```
public static final short SHORT_SMOOTHING_BUFFER
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

STREAM_IDENTIFIER

```
public static final short STREAM_IDENTIFIER
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

STUFFING

```
public static final short STUFFING
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

SUBTITLING

```
public static final short SUBTITLING
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

TELEPHONE

```
public static final short TELEPHONE
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

TELETEXT

```
public static final short TELETEXT
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

TERRESTRIAL_DELIVERY_SYSTEM

```
public static final short TERRESTRIAL_DELIVERY_SYSTEM
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

TIME_SHIFTED_EVENT

```
public static final short TIME_SHIFTED_EVENT
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

TIME_SHIFTED_SERVICE

```
public static final short TIME_SHIFTED_SERVICE
```

Constant value for the descriptor tag as specified in EN 300 468 [4].

org.dvb.si

PMTElementaryStream

Declaration

```
public interface PMTElementaryStream extends SIInformation
```

All Superinterfaces:

```
SIInformation
```

Description

This interface represents an elementary stream of a service.

For each running service there is a PMT describing the elementary streams of the service. An object that implements this interface represents one such elementary stream. Each object that implements the PMTElementaryStream interface is identified by the combination of the identifiers original_network_id, transport_stream_id, service_id, component_tag (or elementary_PID).

See Also:

```
PMTService, PMTStreamType
```

Methods

getComponentTag()

```
public int getComponentTag()
```

Get the component tag identifier.

Returns:

The component tag. If the elementary stream does not have an associated component tag, this method returns -2.

getDvbLocator()

```
public org.davic.net.dvb.DvbLocator getDvbLocator()
```

Gets a DvbLocator that identifies this elementary stream.

Returns:

The DvbLocator of this elementary stream.

getElementaryPID()

```
public short getElementaryPID()
```

Get the elementary PID.

Returns:

The PID the data of elementary stream is sent on in the transport stream.

getOriginalNetworkID()

```
public int getOriginalNetworkID()
```

Get the original network identification identifier.

Returns:

The original network identification.

getServiceID()

```
public int getServiceID()
```

Get the service identification identifier.

Returns:

The service identification.

getStreamType()

```
public byte getStreamType()
```

Get the stream type of this elementary stream. The value returned shall be the actual value from the descriptor loop and is not limited to the set of values defined in `PMTStreamType`.

Returns:

The stream type (some of the possible values are defined in the PMTStreamType interface).

See Also:

`PMTStreamType`

getTransportStreamID()

```
public int getTransportStreamID()
```

Get the transport stream identification identifier.

Returns:

The transport stream identification.

org.dvb.si

PMTService

Declaration

```
public interface PMTService extends SIInformation
```

All Superinterfaces:

`SIInformation`

Description

This interface represents a particular service carried by a transport stream. The information is retrieved from the PMT table.

Each object that implements the PMTService interface is identified by the combination of the following identifiers: original_network_id, transport_stream_id, service_id.

Methods

getDvbLocator()

```
public org.davic.net.dvb.DvbLocator getDvbLocator()
```

Gets a DvbLocator that identifies this service.

Returns:

The DvbLocator of this service.

getOriginalNetworkID()

```
public int getOriginalNetworkID()
```

Get the original network identification.

Returns:

The original network identification identifier.

getPcrPid()

```
public int getPcrPid()
```

Get the PCR pid.

Returns:

The PCR pid.

getServiceID()

```
public int getServiceID()
```

Get the service identification.

Returns:

The service identification identifier.

getTransportStreamID()

```
public int getTransportStreamID()
```

Get the transport stream identification.

Returns:

The transport stream identification identifier.

retrievePMTElementaryStreams(short, Object, SIRetrievalListener, short[])

```
public org.dvb.si.SIRequest retrievePMTElementaryStreams(short retrieveMode,
java.lang.Object appData, org.dvb.si.SIRetrievalListener listener, short[] somePMTDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve information associated with the elementary streams which compose this service from the Program Map Table (PMT).

The SIIterator that is returned with the event when the request completes successfully will contain one or more objects that implement the PMTElementaryStream interface. If no matching object was found, the appropriate one of the following events is sent: SINotInCacheEvent SIObjectNotInTableEvent or SITableNotFoundEvent. This method will retrieve PMTElementaryStreams from the same sub-table version as this PMTService instance. If this version of the sub-table is no longer available, an SITableUpdatedEvent is returned.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`somePMTDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If some-PMTDescriptorTags 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

Declaration

`public interface` **`PMTStreamType`**

Description

This interface defines the constants corresponding to the different stream types.

See Also:

`PMTElementaryStream`, `PMTElementaryStream.getStreamType()`

Fields

MPEG1_AUDIO

`public static final byte` **`MPEG1_AUDIO`**

Constant value for the stream type as specified in ISO/IEC 13818-1 [15].

MPEG1_VIDEO

`public static final byte` **`MPEG1_VIDEO`**

Constant value for the stream type as specified in ISO/IEC 13818-1 [15].

MPEG2_AUDIO

```
public static final byte MPEG2_AUDIO
```

Constant value for the stream type as specified in ISO/IEC 13818-1 [15].

MPEG2_VIDEO

```
public static final byte MPEG2_VIDEO
```

Constant value for the stream type as specified in ISO/IEC 13818-1 [15].

org.dvb.si

SIBouquet

Declaration

```
public interface SIBouquet extends SIInformation
```

All Superinterfaces:

```
SIInformation
```

Description

This interface (together with the SITransportStreamBAT interface) represents a sub-table of the Bouquet Association Table (BAT) describing a particular bouquet.

Each object that implements the SIBouquet interface is identified by the identifier bouquet_id.

See Also:

```
SITransportStreamBAT
```

Methods

getBouquetID()

```
public int getBouquetID()
```

Get the identification.

Returns:

The bouquet identification of this bouquet.

getDescriptorTags()

```
public short[] getDescriptorTags()
```

This method defines extra semantics for the SIInformation.getDescriptorTags method. If the BAT sub-table on which this SIBouquet object is based consists of multiple sections, then this method returns the descriptor tags in the order they appear when concatenating the descriptor loops of the different sections.

Overrides:

`getDescriptorTags` in interface `SIInformation`

Returns:

The tags of the descriptors actually broadcast for the object (identified by their tags).

See Also:

`SIInformation, SIInformation.getDescriptorTags().`

getName()

```
public java.lang.String getName()
```

This method returns the name of this bouquet. If the language returned by `javax.tv.service.SIManager.get-PreferredLanguage` is one of those in the multilingual_bouquet_name_descriptor, return the name in that language, otherwise return an implementation dependent selection between the names in the multilingual_bouquet_name_descriptor and the name in the bouquet_name_descriptor. When this information is not available "" is returned. All control characters as defined in ETR 211 [i.3] 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 [i.3]) of this bouquet without emphasis marks. If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` is one of those in the multilingual_bouquet_name_descriptor, return the name in that language, otherwise return an implementation dependent selection between the names in the multilingual_bouquet_name_descriptor and the name in the bouquet_name_descriptor. If the descriptor is not present, "" is returned. If the string can be found but does not contain control codes for abbreviating it, the full string shall be returned. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

Returns:

The short bouquet name of this bouquet.

getSIServiceLocators()

```
public org.davic.net.dvb.DvbLocator[] getSIServiceLocators()
```

Get a list of DvbLocators identifying the services that belong to the bouquet.

Returns:

An array of DvbLocators identifying the services.

See Also:

`org.davic.net.dvb.DvbLocator, SIService`

retrieveDescriptors(short, Object, SIRetrievalListener)

```
public org.dvb.si.SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener)
throws SIIllegalArgumentException
```

This method defines extra semantics for the SIInformation.retrieveDescriptors method (first prototype). If the BAT sub-table on which this SIBouquet object is based consists of multiple sections, then this method returns the requested descriptors in the order they appear when concatenating the descriptor loops of the different sections.

Overrides:

`retrieveDescriptors` in interface `SIInformation`

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid.

See Also:

`SIInformation, SIInformation.retrieveDescriptors(short, Object, SIRetrievalListener).`

retrieveDescriptors(short, Object, SIRetrievalListener, short[])

```
public org.dvb.si.SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

This method defines extra semantics for the SIInformation.retrieveDescriptors method (second prototype). If the BAT sub-table on which this SIBouquet object is based consists of multiple sections, then this method returns the requested descriptors in the order they appear when concatenating the descriptor loops of the different sections. The tag values included in the someDescriptorsTags parameter are used for filtering the information that is returned. Only information related to descriptors whose tag value is included in the someDescriptorTags array is retrieved, unless the someDescriptorTags array contains -1 as its one and only item.

Overrides:

`retrieveDescriptors` in interface `SIInformation`

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`someDescriptorTags` - Descriptor tag values that are used for filtering descriptors from descriptors included in the SI table item corresponding to this object. If the array contains -1 as its one and only element, all descriptors related to this object are retrieved. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid.

See Also:

`SIInformation, SIInformation.retrieveDescriptors(short, Object, SIRetrievalListener, short[])`

retrieveSIBouquetTransportStreams(short, Object, SIRetrievalListener, short[])

```
public org.dvb.si.SIRequest retrieveSIBouquetTransportStreams(short retrieveMode,
java.lang.Object appData, org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve information associated with transport streams belonging to the bouquet.

The SIIterator that is returned with the event when the request completes successfully will contain one or more objects that implement the SITransportStreamBAT interface. This method will retrieve SIBouquetTransportStreams from the same sub-table version as this SIBouquet instance. If this version of the sub-table is no longer available, an SITable-UpdatedEvent is returned.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptorTags is null, the behaviour is implementation dependent. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid.

See Also:

`SIRequest, SIRetrievalListener, SITransportStreamBAT, DescriptorTag.`

org.dvb.si

SIDatabase

Declaration

```
public class SIDatabase

java.lang.Object
  |
  +--org.dvb.si.SIDatabase
```

Description

This class represents the root of the SI information hierarchy. There is one SIDatabase per network interface. In a system with a single network interface there is only one SIDatabase object.

When adding a listener to monitor for changes in an SI table (or data carried in an SI table), an event shall not be generated for the current version of that table (or data) found in the network at the time the listener is added. Events shall only be generated for changes following the detection of that current version.

Constructors

SIDatabase()

```
protected SIDatabase()
```

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

Methods

addBouquetMonitoringListener(SIMonitoringListener, int)

```
public void addBouquetMonitoringListener(org.dvb.si.SIMonitoringListener listener, int bouquetId)
throws SIIllegalArgumentException
```

Initiate monitoring of the bouquet information. When the bouquet information changes, an event will be delivered to the registered listener object.

How the monitoring is performed is implementation dependent and especially does not necessarily need to be continuous. The event will be delivered as soon as the implementation notices the change which might have some delay relative to when the change was actually made in the stream due to resources for the monitoring being scheduled between the monitoring activities of different tables. The present document does not set any minimum requirements for monitoring of the SI tables. This is to be done at a best effort basis by the implementation and is entirely implementation dependent. The only requirement is that when an implementation detects a change, e.g. because a resident Navigator or an MHP application has retrieved some SI information from the stream, then these listeners are notified of the change.

The monitoring stops silently and permanently when the network interface with which this SIDatabase object is associated starts tuning to another transport stream.

Parameters:

`listener` - listener object that will receive events when a change in the information is detected.

`bouquetId` - bouquet identifier of the bouquet whose information will be monitored.

Throws:

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range).

See Also:

`SIMonitoringListener, SIMonitoringEvent.`

addEventPresentFollowingMonitoringListener(SIMonitoringListener, int, int, int)

```
public void addEventPresentFollowingMonitoringListener(org.dvb.si.SIMonitoringListener listener,
int originalNetworkId, int transportStreamId, int serviceId)
throws SIIllegalArgumentException
```

Initiate monitoring of information in the EIT related to present and following events. When the information related to those events changes, an event will be delivered to the registered listener object.

The scope of the monitoring is determined by the original network identifier, transport stream identifier and service identifier. The listener will be notified about the change of the information in any present and following event within that scope.

How the monitoring is performed is implementation dependent and especially does not necessarily need to be continuous. The event will be delivered as soon as the implementation notices the change which might have some delay relative to when the change was actually made in the stream due to resources for the monitoring being scheduled between the monitoring activities of different tables. The present document does not set any minimum requirements for monitoring of the SI tables. This is to be done at a best effort basis by the implementation and is entirely implementation dependent. The only requirement is that when an implementation detects a change, e.g. because a resident Navigator or an MHP application has retrieved some SI information from the stream, then these listeners are notified of the change.

The monitoring stops silently and permanently when the network interface with which this SIDatabase object is associated starts tuning to another transport stream. When the referenced service is carried in the currently tuned transport stream, the terminal shall monitor this EITp/f actual and report the changes to any registered listener(s).

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(org.dvb.si.SIMonitoringListener listener,
int originalNetworkId, int transportStreamId, int serviceId, java.util.Date startTime,
java.util.Date endTime)
throws SIIllegalArgumentException, SIInvalidPeriodException
```

Initiate monitoring of information in the EIT related to scheduled events. When the information related to those events changes, an event will be delivered to the registered listener object.

The scope of the monitoring is determined by the original network identifier, transport stream identifier, service identifier, start time and end time of the schedule period. The listener will be notified about the change of the information in any scheduled event within that scope. The scope of the start time and end time shall be as specified for the parameters of the same name in SIService.retrieveScheduledSIEvents.

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 present document does not set any minimum requirements for monitoring of the SI tables. This is to be done at a best effort basis by the implementation and is entirely implementation dependent. The only requirement is that when an implementation detects a change, e.g. because a resident Navigator or an MHP application has retrieved some SI information from the stream, then these listeners are notified of the change.

The monitoring stops silently and permanently when the network interface with which this SIDatabase object is associated starts tuning to another transport stream.

Parameters:

`listener` - listener object that will receive events when a change in the information is detected.

`originalNetworkId` - original network identifier specifying the scope of the monitoring.

`transportStreamId` - transport stream identifier specifying the scope of the monitoring.

`serviceId` - service identifier specifying the scope of the monitoring.

`startTime` - start time of the schedule period.

`endTime` - end time of the schedule period.

Throws:

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range).

`SIInvalidPeriodException` - thrown if end time is before start time.

See Also:

`SIMonitoringListener, SIMonitoringEvent.`

addNetworkMonitoringListener(SIMonitoringListener, int)

```
public void addNetworkMonitoringListener(org.dvb.si.SIMonitoringListener listener, int networkId)
throws SIIllegalArgumentException
```

Initiate monitoring of the network information. When the network information changes, an event will be delivered to the registered listener object.

How the monitoring is performed is implementation dependent and especially does not necessarily need to be continuous. The event will be delivered as soon as the implementation notices the change which might have some delay relative to when the change was actually made in the stream due to resources for the monitoring being scheduled between the monitoring activities of different tables. The present document does not set any minimum requirements for monitoring of the SI tables. This is to be done at a best effort basis by the implementation and is entirely implementation dependent. The only requirement is that when an implementation detects a change, e.g. because a resident Navigator or an MHP application has retrieved some SI information from the stream, then these listeners are notified of the change.

The monitoring stops silently and permanently when the network interface with which this SIDatabase object is associated starts tuning to another transport stream.

Parameters:

`listener` - listener object that will receive events when a change in the information is detected.

`networkId` - network identifier of the network whose information will be monitored.

Throws:

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range).

See Also:

`SIMonitoringListener, SIMonitoringEvent.`

addPMTServiceMonitoringListener(SIMonitoringListener, int, int, int)

```
public void addPMTServiceMonitoringListener(org.dvb.si.SIMonitoringListener listener,
int originalNetworkId, int transportStreamId, int serviceId)
throws SIIllegalArgumentException
```

Initiate monitoring of information in the PMT related to a service. When the information related to a service changes, an event will be delivered to the registered listener object.

How the monitoring is performed is implementation dependent and especially does not necessarily need to be continuous. The event will be delivered as soon as the implementation notices the change which might have some delay relative to when the change was actually made in the stream due to resources for the monitoring being scheduled between the monitoring activities of different tables. Except as specified below, the present document does not set any minimum requirements for monitoring of the SI tables. This is to be done at a best effort basis by the implementation and is entirely implementation dependent. The only requirement is that when an implementation detects a change, e.g. because a resident Navigator or an MHP application has retrieved some SI information from the stream, then these listeners are notified of the change. When the referenced service is the currently selected service within a service context, the terminal shall monitor this PMT and report the changes to any registered listener(s).

The monitoring stops silently and permanently when the network interface with which this SIDatabase object is associated starts tuning to another transport stream.

Parameters:

`listener` - listener object that will receive events when a change in the information is detected.

`originalNetworkId` - original network identifier of the service.

`transportStreamId` - transport stream identifier of the service.

`serviceId` - service identifier specifying the service whose information will be monitored.

Throws:

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range).

See Also:

`SIMonitoringListener, SIMonitoringEvent.`

addServiceMonitoringListener(SIMonitoringListener, int, int)

```
public void addServiceMonitoringListener(org.dvb.si.SIMonitoringListener listener,
int originalNetworkId, int transportStreamId)
throws SIIllegalArgumentException
```

Initiate monitoring of information in the SDT related to services. When the information related to services changes, an event will be delivered to the registered listener object.

The scope of the monitoring is determined by the original network identifier and transport stream identifier. The listener will be notified about the change of the information in any service within that scope.

How the monitoring is performed is implementation dependent and especially does not necessarily need to be continuous. The event will be delivered as soon as the implementation notices the change which might have some delay relative to when the change was actually made in the stream due to resources for the monitoring being scheduled between the monitoring activities of different tables. The present document does not set any minimum requirements for monitoring of the SI tables. This is to be done at a best effort basis by the implementation and is entirely implementation dependent. The only requirement is that when an implementation detects a change, e.g. because a resident Navigator or an MHP application has retrieved some SI information from the stream, then these listeners are notified of the change.

The monitoring stops silently and permanently when the network interface with which this SIDatabase object is associated starts tuning to another transport stream.

Parameters:

`listener` - listener object that will receive events when a change in the information is detected.

`originalNetworkId` - original network identifier specifying the scope of the monitoring.

`transportStreamId` - transport stream identifier specifying the scope of the monitoring.

Throws:

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range).

See Also:

`SIMonitoringListener, SIMonitoringEvent.`

getSIDatabase()

```
public static org.dvb.si.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(org.dvb.si.SIMonitoringListener listener, int bouquetId)
throws SIIllegalArgumentException
```

Removes the registration of an event listener for bouquet information monitoring. If this method is called with a listener that is registered but not with the same identifiers of the SI objects as given in the parameters, the method shall fail silently and the listeners stays registered with those identifiers that it has been added.

Parameters:

`listener` - listener object that has previously been registered.

`bouquetId` - bouquet identifier of the bouquet whose information has been requested to be monitored.

Throws:

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range).

See Also:

`SIMonitoringListener, SIMonitoringEvent.`

removeEventPresentFollowingMonitoringListener(SIMonitoringListener, int, int, int)

```
public void removeEventPresentFollowingMonitoringListener(org.dvb.si.SIMonitoringListener listener,
int originalNetworkId, int transportStreamId, int serviceId)
throws SIIllegalArgumentException
```

Removes the registration of an event listener for monitoring information related to present and following events If this method is called with a listener that is registered but not with the same identifiers of the SI objects as given in the parameters, the method shall fail silently and the listeners stays registered with those identifiers that it has been added.

Parameters:

`listener` - listener object that has previously been registered.

`originalNetworkId` - original network identifier specifying the scope of the monitoring.

`transportStreamId` - transport stream identifier specifying the scope of the monitoring.

`serviceId` - service identifier specifying the scope of the monitoring.

Throws:

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range).

See Also:

`SIMonitoringListener, SIMonitoringEvent.`

removeEventScheduleMonitoringListener(SIMonitoringListener, int, int, int)

```
public void removeEventScheduleMonitoringListener(org.dvb.si.SIMonitoringListener listener,
int originalNetworkId, int transportStreamId, int serviceId)
throws SIIllegalArgumentException
```

Removes the registration of an event listener for monitoring information related to scheduled events for all periods If this method is called with a listener that is registered but not with the same identifiers of the SI objects as given in the parameters, the method shall fail silently and the listeners stays registered with those identifiers that it has been added.

Parameters:

`listener` - listener object that has previously been registered.

`originalNetworkId` - original network identifier specifying the scope of the monitoring.

`transportStreamId` - transport stream identifier specifying the scope of the monitoring.

`serviceId` - service identifier specifying the scope of the monitoring.

Throws:

`SIIllegalArgumentException` - thrown if the identifiers are invalid (e.g. out of range).

See Also:

SIMonitoringListener, SIMonitoringEvent.

removeNetworkMonitoringListener(SIMonitoringListener, int)

```
public void removeNetworkMonitoringListener(org.dvb.si.SIMonitoringListener listener, int networkId)
throws SIIllegalArgumentException
```

Removes the registration of an event listener for network information monitoring. If this method is called with a listener that is registered but not with the same identifiers of the SI objects as given in the parameter, the method shall fail silently and the listeners stays registered with those identifiers that it has been added.

Parameters:

listener - listener object that has previously been registered.

networkId - network identifier of the network which is no longer to be monitored by the listener.

Throws:

SIIllegalArgumentException - thrown if the identifiers are invalid (e.g. out of range).

See Also:

SIMonitoringListener, SIMonitoringEvent.

removePMTServiceMonitoringListener(SIMonitoringListener, int, int, int)

```
public void removePMTServiceMonitoringListener(org.dvb.si.SIMonitoringListener listener,
int originalNetworkId, int transportStreamId, int serviceId)
throws SIIllegalArgumentException
```

Removes the registration of an event listener for monitoring information in the PMT related to a service. If this method is called with a listener that is registered but not with the same identifiers of the SI objects as given in the parameters, the method shall fail silently and the listeners stays registered with those identifiers that it has been added.

Parameters:

listener - listener object that has previously been registered.

originalNetworkId - original network identifier of the service.

transportStreamId - transport stream identifier of the service.

serviceId - service identifier specifying the service whose information has been requested to be monitored.

Throws:

SIIllegalArgumentException - thrown if the identifiers are invalid (e.g. out of range).

See Also:

SIMonitoringListener, SIMonitoringEvent.

removeServiceMonitoringListener(SIMonitoringListener, int, int)

```
public void removeServiceMonitoringListener(org.dvb.si.SIMonitoringListener listener,
int originalNetworkId, int transportStreamId)
throws SIIllegalArgumentException
```

Removes the registration of an event listener for monitoring information related to services. If this method is called with a listener that is registered but not with the same identifiers of the SI objects as given in the parameters, the method shall fail silently and the listeners stays registered with those identifiers that it has been added.

Parameters:

listener - listener object that has previously been registered.

originalNetworkId - original network identifier specifying the scope of the monitoring.

transportStreamId - transport stream identifier specifying the scope of the monitoring.

Throws:

SIIllegalArgumentException - thrown if the identifiers are invalid (e.g. out of range).

See Also:

SIMonitoringListener, SIMonitoringEvent.

retrieveActualSINetwork(short, Object, SIRetrievalListener, short[])

```
public org.dvb.si.SIRequest retrieveActualSINetwork(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve information associated with the actual network. The actual network is the network carrying the transport stream currently selected by the network interface connected to this SIDatabase.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SINetwork interface. If no matching object was found, the appropriate one of the following events is sent: SINotInCacheEvent or SITableNotFoundEvent.

Parameters:

retrieveMode - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

appData - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

listener - SIRetrievalListener that will receive the event informing about the completion of the request.

someDescriptorTags - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptorTags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

SIIllegalArgumentException - thrown if the retrieveMode is invalid.

See Also:

SIRequest, SIRetrievalListener, SINetwork, DescriptorTag.

retrieveActualSIServices(short, Object, SIRetrievalListener, short[])

```
public org.dvb.si.SIRequest retrieveActualSIServices(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve information associated with the actual services. The actual services are the services in the transport stream currently selected by the network interface connected to this SIDatabase.

The SIIterator that is returned with the event when the request completes successfully will contain one or more objects that implement the SIService interface. If no matching object was found, the appropriate one of the following events is sent: SINotInCacheEvent, SIObjectNotInTableEvent or SITableNotFoundEvent.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptor-Tags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid.

See Also:

`SIRequest, SIRetrievalListener, SIService, DescriptorTag.`

retrieveActualSITransportStream(short, Object, SIRetrievalListener, short[])

```
public org.dvb.si.SIRequest retrieveActualSITransportStream(short retrieveMode,
java.lang.Object appData, org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve information associated with the actual transport stream. The actual transport stream is the transport stream currently selected by the network interface connected to this SIDatabase.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SITransportStreamNIT interface. If no matching object was found, the appropriate one of the following events is sent: SINotInCacheEvent SIObjectNotInTableEvent or SITableNotFoundEvent.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptor-Tags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid.

See Also:

SIRequest, SIRetrievalListener, SITransportStream, DescriptorTag.

retrievePMTElementaryStreams(short, Object, SIRetrievalListener, DvbLocator, short[])

```
public org.dvb.si.SIRequest retrievePMTElementaryStreams(short retrieveMode,
java.lang.Object appData, org.dvb.si.SIRetrievalListener listener,
org.davic.net.dvb.DvbLocator dvbLocator, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve PMT elementary stream information associated with components of a service. The required component(s) can be specified by its DVB locator.

The SIIterator that is returned with the event when the request completes successfully will contain one or more objects that implement the PMTElementaryStream interface. If no matching object was found, the appropriate one of the following events is sent: SINotInCacheEvent, SIObjectNotInTableEvent or SITableNotFoundEvent.

Parameters:

retrieveMode - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

appData - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

listener - SIRetrievalListener that will receive the event informing about the completion of the request.

dvbLocator - DVB Locator identifying the 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 someDescriptor-Tags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

SIIllegalArgumentException - thrown if the retrieveMode is invalid or if the locator is invalid and does not identify one or more service components.

See Also:

SIRequest, SIRetrievalListener, SIService, DescriptorTag.

retrievePMTElementaryStreams(short, Object, SIRetrievalListener, int, int, short[])

```
public org.dvb.si.SIRequest retrievePMTElementaryStreams(short retrieveMode,
java.lang.Object appData, org.dvb.si.SIRetrievalListener listener, int serviceId, int componentTag,
short[] someDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve PMT elementary stream information associated with components of a service from the actual transport stream of this SIDatabase object. The elementary streams can be specified by their identification. When -1 is specified for componentTag then elementary streams shall be retrieved regardless of their component tag.

The SIIterator that is returned with the event when the request completes successfully will contain one or more objects that implement the PMTElementaryStream interface. If no matching object was found, the appropriate one of the following events is sent: SINotInCacheEvent, SIObjectNotInTableEvent or SITableNotFoundEvent.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`serviceId` - Identification of the elementary streams to be retrieved: service identifier.

`componentTag` - Identification of the elementary streams to be retrieved: component tag (-1 means return elementary streams regardless of their component tag).

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptor-Tags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid or the numeric identifiers are out of range.

See Also:

`SIRequest, SIRetrievalListener, SIService, DescriptorTag.`

retrievePMTService(short, Object, SIRetrievalListener, DvbLocator, short[])

```
public org.dvb.si.SIRequest retrievePMTService(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener, org.davic.net.dvb.DvbLocator dvbLocator,
short[] someDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve PMT information associated with a service. The required service can be specified by its DVB locator.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the PMTService interface. If no matching object was found, the appropriate one of the following events is sent: SINotInCacheEvent SIObjectNotInTableEvent or SITableNotFoundEvent.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`dvbLocator` - DVB Locator identifying the service. The locator may be more specific than identifying a service, but this method will only use the parts starting from the beginning up to the service id.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptor-Tags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid or the locator is invalid and does not identify a service.

See Also:

`SIRequest, SIRetrievalListener, SIService, DescriptorTag.`

retrievePMTServices(short, Object, SIRetrievalListener, int, short[])

```
public org.dvb.si.SIRequest retrievePMTServices(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener, int serviceId, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve PMT information associated with services from the actual transport stream of this SIDatabase object. The required services can be specified by their identification. When -1 is specified as `serviceId` then services shall be retrieved regardless of their service id.

The SIIterator that is returned with the event when the request completes successfully will contain one or more objects that implement the PMTService interface. If no matching object was found, the appropriate one of the following events is sent: `SINotInCacheEvent`, `SIObjectNotInTableEvent` or `SITableNotFoundEvent`.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`serviceId` - Identification of the services to be retrieved: service identifier (-1 means return services regardless of their service id).

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptor-Tags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid or the numeric identifiers are out of range.

See Also:

`SIRequest, SIRetrievalListener, SIService, DescriptorTag.`

retrieveSIBouquets(short, Object, SIRetrievalListener, int, short[])

```
public org.dvb.si.SIRequest retrieveSIBouquets(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener, int bouquetId, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve information associated with bouquets. A bouquet can be specified by its identification. When bouquetId is set to -1, all bouquets signalled in the BAT of the currently received transport stream on that network interface are retrieved.

The SIIterator that is returned with the event when the request completes successfully will contain one or more objects that implement the SIBouquet interface. If no matching object was found, the appropriate one of the following events is sent: `SINotInCacheEvent`, `SIObjectNotInTableEvent` or `SITableNotFoundEvent`.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`bouquetId` - Identifier of the bouquet to be retrieved or -1 for all bouquets signalled on the currently received transport stream.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptor-Tags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid or the numeric identifiers are out of range.

See Also:

`SIRequest, SIRetrievalListener, SIBouquet, DescriptorTag.`

retrieveSINetworks(short, Object, SIRetrievalListener, int, short[])

```
public org.dvb.si.SIRequest retrieveSINetworks(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener, int networkId, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve information associated with networks. A network can be specified by its identification. When networkId is set to -1, all networks signalled in NIT Actual and Other of the currently received TransportStream on that network interface shall be retrieved.

The SIIterator that is returned with the event when the request completes successfully will contain one or more objects that implement the SINetwork interface. If no matching object was found, the appropriate one of the following events is sent: `SINotInCacheEvent`, `SITableNotFoundEvent`.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

networkId - Identification of the network to be retrieved or -1 for all networks currently signalled.

someDescriptorTags - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptor-Tags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

SIIllegalArgumentException - thrown if the retrieveMode is invalid or the numeric identifiers are out of range.

See Also:

SIRequest, SIRetrievalListener, SINetwork, DescriptorTag.

retrieveSIService(short, Object, SIRetrievalListener, DvbLocator, short[])

```
public org.dvb.si.SIRequest retrieveSIService(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener, org.davic.net.dvb.DvbLocator dvbLocator,
short[] someDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve information associated with a service. The required service can be specified by its DVB locator.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SIService interface. If no matching object was found, the appropriate one of the following events is sent: SINotInCacheEvent SIObjectNotInTableEvent or SITableNotFoundEvent".

Parameters:

retrieveMode - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

appData - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

listener - SIRetrievalListener that will receive the event informing about the completion of the request.

dvbLocator - DVB locator identifying the service.The locator may be more specific than identifying a service, but this method will only use the parts starting from the beginning up to the service id.

someDescriptorTags - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptor-Tags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

SIIllegalArgumentException - thrown if the retrieveMode is invalid or the locator is invalid and does not identify a service.

See Also:

SIRequest, SIRetrievalListener, SIService, DescriptorTag.

retrieveSIServices(short, Object, SIRetrievalListener, int, int, int, short[])

```
public org.dvb.si.SIRequest retrieveSIServices(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener, int originalNetworkId, int transportStreamId,
int serviceId, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve information associated with services. The required services can be specified by their identification. When -1 is specified for `transportStreamId` then services shall be retrieved regardless of their transport stream id. When -1 is specified for `serviceId` then services shall be retrieved regardless of their service id.

The SIIterator that is returned with the event when the request completes successfully will contain one or more objects that implement the SIService interface. If no matching object was found, the appropriate one of the following events is sent: `SINotInCacheEvent`, `SIObjectNotInTableEvent` or `SITableNotFoundEvent`.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`originalNetworkId` - Identification of the services to be retrieved: original network identifier.

`transportStreamId` - Identification of the services to be retrieved: transport stream identifier (-1 means return services regardless of their transport stream id).

`serviceId` - Identification of the services to be retrieved: service identifier (-1 means return services regardless of their service id).

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptor-Tags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid or the numeric identifiers are out of range.

See Also:

`SIRequest, SIRetrievalListener, SIService, DescriptorTag.`

retrieveSITimeFromTDT(short, Object, SIRetrievalListener)

```
public org.dvb.si.SIRequest retrieveSITimeFromTDT(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener)
throws SIIllegalArgumentException
```

Retrieve information associated with time from the Time and Date Table (TDT) from the actual transport stream.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SITime interface. If no matching object was found, the appropriate one of the following events is sent: SINotInCacheEvent SIObjectNotInTableEvent or SITableNotFoundEvent.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid.

See Also:

`SIRequest, SIRetrievalListener, SITime.`

retrieveSITimeFromTOT(short, Object, SIRetrievalListener, short[])

```
public org.dvb.si.SIRequest retrieveSITimeFromTOT(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve information associated with time from the Time Offset Table (TOT) from the actual transport stream. The time information will be accompanied with offset information.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SITime interface. If no matching object was found, the appropriate one of the following events is sent: SINotInCacheEvent SIObjectNotInTableEvent or SITableNotFoundEvent.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptor-Tags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid.

See Also:

`SIRequest, SIRetrievalListener, SITime.`

retrieveSITransportStreamDescription(short, Object, SIRetrievalListener, short[])

```
public org.dvb.si.SIRequest retrieveSITransportStreamDescription(short retrieveMode,
java.lang.Object appData, org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve the SITransportStreamDescription object representing the information of the TSDT table in the actual transport stream of this SIDatabase object.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SITransportStreamDescription interface. If no matching object was found, the appropriate one of the following events is sent: SINotInCacheEvent SIObjectNotInTableEvent or SITableNotFoundEvent.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptor-Tags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid.

See Also:

`SIRequest, SIRetrievalListener, SITransportStreamDescription, DescriptorTag.`

org.dvb.si

SIEvent

Declaration

```
public interface SIEvent extends SIInformation
```

All Superinterfaces:

`SIInformation`

Description

This interface represents a particular event within a service.

Each object that implements the SIEvent interface is defined by the combination of the identifiers original_network_id, transport_stream_id, service_id, event_id.

Where methods return values from a short_event_descriptor, the following algorithm shall be used where more than one such descriptor is present. If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` is one of those for which there is a short_event_descriptor then return the value from that descriptor. Otherwise return an implementation dependent selection between the values in the available short_event_descriptors.

See Also:

`SIService`

Methods

getContentNibbles()

```
public byte[] getContentNibbles()
```

This method returns the content nibbles related to the event. This information is extracted from the content_descriptor. If this descriptor is not present an empty array is returned (array with length 0). The return value is an array, each array element describes one content nibble. In each nibble the level 1 content nibbles occupy the four most significant bits of the returned bytes, level 2 content nibbles the four least significant bits.

Returns:

The content nibbles related to the event; level 1 content nibbles occupy the four most significant bits of the returned bytes, level 2 content nibbles the four least significant bits.

getDuration()

```
public long getDuration()
```

Get the duration of this event.

Returns:

The duration in seconds.

getDvbLocator()

```
public org.davic.net.dvb.DvbLocator getDvbLocator()
```

Gets a DvbLocator that identifies this event.

Returns:

The DvbLocator of this event.

getEventID()

```
public int getEventID()
```

Get the event identification.

Returns:

The event identification.

getFreeCAMode()

```
public boolean getFreeCAMode()
```

Get the free_CA_mode value for this event, false indicates none of the component streams of this event are scrambled.

Returns:

The free_CA_mode value.

getLevel1ContentNibbles()

```
public byte[] getLevel1ContentNibbles()
```

This method returns the level 1 content nibbles of this event. This information is extracted from the content_descriptor. If this descriptor is not present an empty array is returned (array with length 0). The return value is an array, each array element describes one content nibble. In each nibble the data occupies the four least significant bits of the returned bytes with the four most significant bits set to 0.

Returns:

All level 1 content nibbles related to the event.

getName()

```
public java.lang.String getName()
```

This method returns the name of this event. The name is extracted from a short_event_descriptor. When this information is not available "" is returned. All control characters as defined in ETR 211 [i.3] 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 [i.3]) of this event without emphasis marks. The name is extracted from a short_event_descriptor. If the descriptor is not present, "" is returned. If the string can be found but does not contain control codes for abbreviating it, the full string shall be returned. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

Returns:

The short event name of this event.

getStartTime()

```
public java.util.Date getStartTime()
```

Get the start time of this event in UTC time.

Returns:

The start time of this event.

getTransportStreamID()

```
public int getTransportStreamID()
```

Get the transport stream identification identifier.

Returns:

The transport stream identification.

retrieveSIService(short, Object, SIRetrievalListener, short[])

```
public org.dvb.si.SIRequest retrieveSIService(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

This method retrieves the SIService object representing the service the event, represented by this SIEvent, is part of.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SIService interface. If no matching object was found, the appropriate one of the following events is sent: SINotInCacheEvent SIObjectNotInTableEvent or SITableNotFoundEvent.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptor-Tags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid.

See Also:

`SIRequest, SIRetrievalListener, SIService, DescriptorTag.`

org.dvb.si

SIException

Declaration

```
public abstract class SIException extends java.lang.Exception
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.dvb.si.SIException
```

All Implemented Interfaces:

```
java.io.Serializable
```

Direct Known Subclasses:

```
SIIllegalArgumentException, SIInvalidPeriodException
```

Description

This class is the root of the SI exceptions hierarchy.

Constructors

SIException()

```
public SIException()
```

Default constructor for the exception.

SIException(String)

```
public SIException(java.lang.String reason)
```

Constructor for the SI exception with a specified reason.

Parameters:

`reason` - the reason why the exception was raised.

org.dvb.si

SIIllegalArgumentException

Declaration

```
public class SIIllegalArgumentException extends SIException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.dvb.si.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()
```

Default constructor for the exception.

SIIllegalArgumentException(String)

```
public SIIllegalArgumentException(java.lang.String reason)
```

Constructor for the exception with a specified reason.

Parameters:

`reason` - the reason why the exception was raised.

org.dvb.si

SIInformation

Declaration

```
public interface SIInformation
```

All Known Subinterfaces:

`PMTElementaryStream`, `PMTService`, `SIBouquet`, `SIEvent`, `SINetwork`, `SIService`, `SITime`, `SITransportStream`, `SITransportStreamBAT`, `SITransportStreamDescription`, `SITransportStreamNIT`

Description

This interface groups the common features of SIBouquet, SINetwork, SITransportStream, SIService, PMTService, SIEvent, SITime and PMTElementaryStream.

Each SIInformation instance represents a sub-table (part). Any method accessing descriptors will retrieve descriptors from the same sub-table version as the SIInformation instance. When this version is no longer available, an SITable-UpdatedEvent is returned.

See Also:

`SIBouquet`, `SINetwork`, `SITransportStream`, `SIService`, `PMTService`, `SIEvent`, `SITime`, `PMTElementaryStream`

Fields

FROM_CACHE_ONLY

```
public static final short FROM_CACHE_ONLY
```

Constant for retrieve mode parameter of the retrieve methods. When FROM_CACHE_ONLY mode is specified, the data will be retrieved only if it is in the cache. Otherwise, SINotInCacheEvent will be delivered to the listener. No stream access is done in this case.

FROM_CACHE_OR_STREAM

```
public static final short FROM_CACHE_OR_STREAM
```

Constant for retrieve mode parameter of the retrieve methods. When FROM_CACHE_OR_STREAM mode is specified, the data will be retrieved from cache if it is present in the cache, otherwise it will be retrieved from the stream.

FROM_STREAM_ONLY

```
public static final short FROM_STREAM_ONLY
```

Constant for retrieve mode parameter of the retrieve methods. When FROM_STREAM_ONLY mode is specified, the data will be retrieved directly from the stream and no cache access is tried first. This mode is meaningful only if the application knows that the information is not in the cache or that the information in the cache is no longer valid, but the implementation of the SI database may not be aware of the invalidity of the cached data. If the application has got the notification of the existence of an updated version through the listener mechanism in this API, the implementation of the SI database is aware of the version change and the application should specify the FROM_CACHE_OR_STREAM mode to be able to retrieve the data faster if the updated version has already been loaded to the cache by the SI database implementation.

Methods

fromActual()

```
public boolean fromActual()
```

Return true when the information contained in the object that implements this interface was filtered from an "actual" table or from a table with no "actual/other" distinction.

Returns:

`true` if the information comes from an "actual" table or from a table with no "actual/other" distinction, otherwise returns `false`.

getDataSource()

```
public org.davic.mpeg.TransportStream getDataSource()
```

Return the org.davic.mpeg.TransportStream object the information contained in the object that implements that interface was filtered from.

Returns:

The `org.davic.mpeg.TransportStream` object the information was filtered from.

See Also:

`org.davic.mpeg.TransportStream.`

getDescriptorTags()

```
public short[] getDescriptorTags()
```

Get the tags of all descriptors that are part of this version of this object. The tags are returned in the same order as the descriptors are broadcast. This method returns also the tags of descriptors that were not hinted at and that are not necessarily present in the cache. If there are no descriptors associated with this SIInformation object, this method returns an empty array whose length is 0.

Returns:

The tags of the descriptors actually broadcast for the object (identified by their tags).

See Also:

`DescriptorTag.`

getSIDatabase()

```
public org.dvb.si.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 org.dvb.si.SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener)
throws SIIllegalArgumentException
```

This method retrieves all descriptors in the order the descriptors are broadcast.

This method is asynchronous and the completion of the method will be signalled by an `SISuccessfulRetrieve-Event` being sent to listener. Any retrieved descriptors are found in the SIIterator returned by the `getResult` method of that event. If descriptors are found then this iterator will contain Descriptor objects. If there are no matching descriptors, this iterator will contain no objects.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid.

See Also:

`SIRequest, SIRetrievalListener, Descriptor, SIIterator.`

retrieveDescriptors(short, Object, SIRetrievalListener, short[])

```
public org.dvb.si.SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve a set of descriptors. This method retrieves all or a set of descriptors in the order the descriptors are broadcast.

The tag values included in the `someDescriptorTags` parameter are used for filtering the descriptors that are returned. Only those descriptors whose tag value is included in the `someDescriptorTags` array are retrieved, unless the `some-DescriptorTags` array contains -1 as its one and only item in which case all descriptors related to this object are retrieved.

If the list of tags is a subset of the one hinted to the underlying implementation (in the request which created the object on which the method is called), this is likely to increase the efficiency of the (optional) caching mechanism.

This method is asynchronous and the completion of the method will be signalled by an `SISuccessfulRetrieve-Event` being sent to listener. Any retrieved descriptors are found in the SIIterator returned by the `getResult` method of that event. If descriptors are found then this iterator will contain Descriptor objects. If there are no matching descriptors, this iterator will contain no objects.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`someDescriptorTags` - Descriptor tag values of descriptors that are used for filtering descriptors from the descriptors included in the SI table item corresponding to this SIInformation object. If the array contains -1 as its one and only element, all descriptors related to this object are retrieved. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid.

See Also:

`SIRequest, SIRetrievalListener, Descriptor, SIIterator, DescriptorTag.`

org.dvb.si

SIInvalidPeriodException

Declaration

```
public class SIInvalidPeriodException extends SIException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.dvb.si.SIException
                    |
                    +--org.dvb.si.SIInvalidPeriodException
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

This exception is thrown when a specified period is invalid (for example, start time is after the end time).

Constructors

SIInvalidPeriodException()

```
public SIInvalidPeriodException()
```

Default constructor for the exception.

SIInvalidPeriodException(String)

```
public SIInvalidPeriodException(java.lang.String reason)
```

Constructor for the exception with a specified reason.

Parameters:

`reason` - the reason why the exception was raised.

org.dvb.si

SIIterator

Declaration

```
public interface SIIterator extends java.util.Enumeration
```

All Superinterfaces:

```
java.util.Enumeration
```

Description

Objects implementing SIIterator interface allow to browse through a set of SI objects. In order to maintain consistency within the set of SI objects, this browsing does NOT initiate an actual access to the stream.

Methods

numberOfRemainingObjects()

```
public int numberOfRemainingObjects()
```

Get the number of remaining objects in the iterator.

Returns:

The number of remaining objects.

org.dvb.si

SILackOfResourcesEvent

Declaration

```
public class SILackOfResourcesEvent extends SIRetrievalEvent
```

```
java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.si.SIRetrievalEvent
              |
              +--org.dvb.si.SILackOfResourcesEvent
```

All Implemented Interfaces:

`java.io.Serializable`

Description

This event is sent in response to a SI retrieval request when the resources needed for retrieving the data are not available, e.g. due to the necessary resources being all taken up by the calling application or other applications.

See Also:

`SIRetrievalListener`

Constructors

SILackOfResourcesEvent(Object, SIRequest)

```
public SILackOfResourcesEvent(java.lang.Object appData, org.dvb.si.SIRequest request)
```

The constructor for the event.

Parameters:

`appData` - the application data passed in the request method call.

`request` - the SIRequest instance which is the source of the event.

org.dvb.si

SIMonitoringEvent

Declaration

```
public class SIMonitoringEvent extends java.util.EventObject
```

```
java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.si.SIMonitoringEvent
```

All Implemented Interfaces:

`java.io.Serializable`

Description

Objects of this class are sent to listener objects of the using application to notify that a change in the monitored information has happened.

See Also:

`SIMonitoringType`, `SIMonitoringListener`

Constructors

SIMonitoringEvent(SIDatabase, byte, int, int, int, int, int, Date, Date)

```
public SIMonitoringEvent(org.dvb.si.SIDatabase source, byte objectType, int networkId,
int bouquetId, int originalNetworkId, int transportStreamId, int serviceId,
java.util.Date startTime, java.util.Date endTime)
```

Constructor for the event object.

Parameters:

`source` - the SIDatabase object which is the source of the event.

`objectType` - type of the SIInformation object (constants in SIMonitoringType).

`networkId` - networkId.

`bouquetId` - bouquetId.

`originalNetworkId` - originalNetworkId.

`transportStreamId` - transportStreamId.

`serviceId` - serviceId.

`startTime` - start time of event schedule period.

`endTime` - end time of event schedule period.

Methods

getBouquetID()

```
public int getBouquetID()
```

Returns the bouquetId of the bouquet. This method is only applicable if the SIInformation type returned with the get-SIInformationType 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 get-SIInformationType 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:

getSource in class EventObject.

Returns:

The SIDatabase instance that is the source of this event.

getStartTime()

```
public java.util.Date getStartTime()
```

Returns the start time of the schedule period whose event information has changed. This method is only applicable if the SIInformation type returned with the getSIInformationType method is SCHEDULED_EVENT.

Returns:

The start time or null if not applicable for this event.

getTransportStreamID()

```
public int getTransportStreamID()
```

Returns the transportStreamId of the SIInformation objects This method is only applicable if the SIInformation type returned with the getSIInformationType method is SERVICE, PMT_SERVICE, PRESENT_FOLLOWING_EVENT or SCHEDULED_EVENT.

Returns:

The transportStreamId or -2 if not applicable for this event.

org.dvb.si SIMonitoringListener

Declaration

```
public interface SIMonitoringListener extends java.util.EventListener
```

All Superinterfaces:

```
java.util.EventListener
```

Description

This interface shall be implemented by using application classes in order to listen to changes in monitored SI objects.

See Also:

```
SIMonitoringEvent
```

Methods

postMonitoringEvent(SIMonitoringEvent)

```
public void postMonitoringEvent(org.dvb.si.SIMonitoringEvent anEvent)
```

This method is called back by the SI API implementation to notify the listener about an event.

Parameters:

`anEvent` - The notified event.

See Also:

`SIMonitoringEvent`

org.dvb.si SIMonitoringType

Declaration

```
public interface SIMonitoringType
```

Description

This interface defines the constants corresponding to the SI information type values in SIMonitoringEvent.

See Also:

`SIMonitoringListener`, `SIMonitoringEvent`

Fields

BOUQUET

```
public static final byte BOUQUET
```

Constant for the type of SIInformation object: Bouquet.

NETWORK

```
public static final byte NETWORK
```

Constant for the type of SIInformation object: Network.

PMT_SERVICE

```
public static final byte PMT_SERVICE
```

Constant for the type of SIInformation object: PMTService.

PRESENT_FOLLOWING_EVENT

```
public static final byte PRESENT_FOLLOWING_EVENT
```

Constant for the type of SIInformation object: Present or following event.

SCHEDULED_EVENT

```
public static final byte SCHEDULED_EVENT
```

Constant for the type of SIInformation object: Scheduled event.

SERVICE

```
public static final byte SERVICE
```

Constant for the type of SIInformation object: Service.

org.dvb.si

SINetwork

Declaration

```
public interface SINetwork extends SIInformation
```

All Superinterfaces:

`SIInformation`

Description

This interface (together with the SITransportStreamNIT interface) represents a sub-table of the Network Information Table (NIT) describing a particular network.

Each object that implements the SINetwork interface is identified by the identifier network_id.

See Also:

`SITransportStream`, `SITransportStreamNIT`

Methods

getDescriptorTags()

```
public short[] getDescriptorTags()
```

This method defines extra semantics for the SIInformation.getDescriptorTags method. If the NIT sub-table on which this SINetwork object is based consists of multiple sections, then this method returns the descriptor tags in the order they appear when concatenating the descriptor loops of the different sections.

Overrides:

`getDescriptorTags` in interface `SIInformation`

Returns:

The tags of the descriptors actually broadcast for the object (identified by their tags).

See Also:

`SIInformation, SIInformation.getDescriptorTags()`

getName()

```
public java.lang.String getName()
```

This method returns the name of this network. If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` is one of those in the multilingual_network_name_descriptor, return the name in that language, otherwise return an implementation dependent selection between the names in the multilingual_network_name_descriptor and the name in the network_name_descriptor. When this information is not available "" is returned. All control characters as defined in ETR 211 [i.3] 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 [i.3]) of this network without emphasis marks. If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` is one of those in the multilingual_network_name_descriptor, return the name in that language, otherwise return an implementation dependent selection between the names in the multilingual_network_name_descriptor and the name in the network_name_descriptor. If the descriptor is not present, "" is returned. If the string can be found but does not contain control codes for abbreviating it, the full string shall be returned. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

Returns:

The short network name of this network.

retrieveDescriptors(short, Object, SIRetrievalListener)

```
public org.dvb.si.SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener)
throws SIIllegalArgumentException
```

This method defines extra semantics for the SIInformation.retrieveDescriptors method (first prototype). If the NIT sub-table on which this SINetwork object is based consists of multiple sections, then this method returns the requested descriptors in the order they appear when concatenating the descriptor loops of the different sections.

Overrides:

`retrieveDescriptors` in interface `SIInformation`

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

appData - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

listener - SIRetrievalListener that will receive the event informing about the completion of the request.

Returns:

An SIRequest object.

Throws:

SIIllegalArgumentException - thrown if the retrieveMode is invalid.

See Also:

```
SIInformation, SIInformation.retrieveDescriptors(short, Object,
    SIRetrievalListener)
```

retrieveDescriptors(short, Object, SIRetrievalListener, short[])

```
public org.dvb.si.SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

This method defines extra semantics for the SIInformation.retrieveDescriptors method (second prototype). If the NIT sub-table on which this SINetwork object is based consists of multiple sections, then this method returns the requested descriptors in the order they appear when concatenating the descriptor loops of the different sections. The tag values included in the someDescriptorsTags parameter are used for filtering the information that is returned. Only information related to descriptors whose tag value is included in the someDescriptorTags array is retrieved, unless the someDescriptorTags array contains -1 as its one and only item.

Overrides:

retrieveDescriptors in interface SIInformation

Parameters:

retrieveMode - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

appData - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

listener - SIRetrievalListener that will receive the event informing about the completion of the request.

someDescriptorTags - Descriptor tag values that are used for filtering descriptors from descriptors included in the SI table item corresponding to this object. If the array contains -1 as its one and only element, all descriptors related to this object are retrieved. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

SIIllegalArgumentException - thrown if the retrieveMode is invalid.

See Also:

```
SIInformation, SIInformation.retrieveDescriptors(short, Object,
SIRetrievalListener, short[])
```

retrieveSITransportStreams(short, Object, SIRetrievalListener, short[])

```
public org.dvb.si.SIRequest retrieveSITransportStreams(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve information associated with transport streams carried via the network.

The SIIterator that is returned with the event when the request completes successfully will contain one or more objects that implement the SITransportStreamNIT interface. This method will retrieve SITransportStreams from the same sub-table version as this SINetwork instance. If this version of the sub-table is no longer available, an SITableUpdated-Event is returned.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptor-Tags is null, the behaviour is implementation dependent. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid.

See Also:

`SIRequest, SIRetrievalListener, SITransportStreamNIT, DescriptorTag`

org.dvb.si

SINotInCacheEvent

Declaration

```
public class SINotInCacheEvent extends SIRetrievalEvent
```

```
java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.si.SIRetrievalEvent
              |
              +--org.dvb.si.SINotInCacheEvent
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

This event is sent in response to a SI retrieval request when the request was made with the FROM_CACHE_ONLY mode and the requested data is not present in the cache.

See Also:

`SIRetrievalListener`

Constructors

SINotInCacheEvent(Object, SIRequest)

```
public SINotInCacheEvent(java.lang.Object appData, org.dvb.si.SIRequest request)
```

The constructor for the event.

Parameters:

`appData` - the application data passed in the request method call.

`request` - the SIRequest instance which is the source of the event.

org.dvb.si

SIObjectNotInTableEvent

Declaration

```
public class SIObjectNotInTableEvent extends SIRetrievalEvent
```

```
java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.si.SIRetrievalEvent
              |
              +--org.dvb.si.SIObjectNotInTableEvent
```

All Implemented Interfaces:

`java.io.Serializable`

Description

This event is sent in response to a SI retrieval request when the SI table where the information about the requested object should be located has been retrieved but the requested object is not present in it. The reason may be that the object corresponding to the requested identifiers does not exist.

See Also:

`SIRetrievalListener`

Constructors

SIObjectNotInTableEvent(Object, SIRequest)

```
public SIObjectNotInTableEvent(java.lang.Object appData, org.dvb.si.SIRequest request)
```

The constructor for the event.

Parameters:

`appData` - the application data passed in the request method call.

`request` - the SIRequest instance which is the source of the event.

org.dvb.si

SIRequest

Declaration

```
public class SIRequest

java.lang.Object
  |
  +--org.dvb.si.SIRequest
```

Description

Object instances of this class represent SI retrieval requests made by the application. The application may cancel the request using this object.

Constructors

SIRequest()

```
protected SIRequest()
```

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

Methods

cancelRequest()

```
public boolean cancelRequest()
```

Cancels the retrieval request.

Returns:

True if the request was cancelled and an SIRequestCancelledEvent will be delivered to the listener, false if the request has already completed (either successfully, with an error or due to a prior cancel method call).

isAvailableInCache()

```
public boolean isAvailableInCache()
```

Returns whether the information will be returned from cache or from the stream.

Returns:

True if the information is available in the cache and will be returned from there otherwise false.

org.dvb.si

SIRequestCancelledEvent

Declaration

```
public class SIRequestCancelledEvent extends SIRetrievalEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.si.SIRetrievalEvent
              |
```

```
            +--org.dvb.si.SIRequestCancelledEvent
```

All Implemented Interfaces:

`java.io.Serializable`

Description

This event is sent in response to a SI retrieval request when the request is cancelled with the SIRequest.cancelRequest method call.

See Also:

`SIRequest`, `SIRetrievalListener`

Constructors

SIRequestCancelledEvent(Object, SIRequest)

```
public SIRequestCancelledEvent(java.lang.Object appData, org.dvb.si.SIRequest request)
```

The constructor for the event.

Parameters:

`appData` - the application data passed in the request method call.

`request` - the SIRequest instance which is the source of the event.

org.dvb.si

SIRetrievalEvent

Declaration

```
public abstract class SIRetrievalEvent extends java.util.EventObject
```

```
java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.si.SIRetrievalEvent
```

All Implemented Interfaces:

`java.io.Serializable`

Direct Known Subclasses:

`SILackOfResourcesEvent`, `SINotInCacheEvent`, `SIObjectNotInTableEvent`, `SIRequestCancelledEvent`, `SISuccessfulRetrieveEvent`, `SITableNotFoundEvent`, `SITableUpdatedEvent`

Description

This class is the base class for events about completion of a SI retrieval request. Exactly one event will be returned in response to an SI retrieval request.

See Also:

`SIRetrievalListener`

Constructors

SIRetrievalEvent(Object, SIRequest)

```
public SIRetrievalEvent(java.lang.Object appData, org.dvb.si.SIRequest request)
```

The constructor for the event.

Parameters:

`appData` - the application data passed in the request method call.

`request` - the SIRequest instance which is the source of the event.

Methods

getAppData()

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

Returns the application data that was passed to the retrieve method.

Returns:

The application data.

getSource()

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

Returns the SIRequest object that is the source of this event.

Overrides:

`getSource` in class `EventObject`

Returns:

The SIRequest object.

org.dvb.si

SIRetrievalListener

Declaration

```
public interface SIRetrievalListener extends java.util.EventListener
```

All Superinterfaces:

```
java.util.EventListener
```

Description

This interface shall be implemented by application classes in order to receive events about completion of SI requests.

See Also:

```
SIRetrievalEvent
```

Methods

postRetrievalEvent(SIRetrievalEvent)

```
public void postRetrievalEvent(org.dvb.si.SIRetrievalEvent event)
```

This method is called by the SI API implementation to notify the listener about completion of an SI request.

Parameters:

`event` - The event object.

See Also:

`SIRetrievalEvent`

org.dvb.si
SIRunningStatus

Declaration

```
public interface SIRunningStatus
```

Description

This interface defines the constants corresponding to the running status values for services and events.

Fields

NOT_RUNNING

```
public static final byte NOT_RUNNING
```

Constant value for the running status as specified in EN 300 468 [4].

PAUSING

```
public static final byte PAUSING
```

Constant value for the running status as specified in EN 300 468 [4].

RUNNING

```
public static final byte RUNNING
```

Constant value for the running status as specified in EN 300 468 [4].

STARTS_IN_A_FEW_SECONDS

```
public static final byte STARTS_IN_A_FEW_SECONDS
```

Constant value for the running status as specified in EN 300 468 [4].

UNDEFINED

```
public static final byte UNDEFINED
```

Constant value for the running status as specified in EN 300 468 [4].

org.dvb.si
SIService

Declaration

```
public interface SIService extends SIInformation, TextualServiceIdentifierQuery
```

All Superinterfaces:

`SIInformation`, `TextualServiceIdentifierQuery`

Description

This interface represents a particular service carried by a transport stream. Information that can be obtained through the methods of this interface is retrieved from the SDT table.

Each object that implements the SIService interface is identified by the combination of the following identifiers: original_network_id, transport_stream_id, service_id.

Methods

getDvbLocator()

```
public org.davic.net.dvb.DvbLocator getDvbLocator()
```

Gets a DvbLocator that identifies this service.

Returns:

The DvbLocator of this service.

getEITPresentFollowingFlag()

```
public boolean getEITPresentFollowingFlag()
```

Get the EIT_present_following_flag value, true indicates this service has present and/or following event information.

Returns:

The EIT_present_following_flag value.

getEITScheduleFlag()

```
public boolean getEITScheduleFlag()
```

Get the EIT_schedule_flag value, true indicates this services has scheduled event information.

Returns:

The EIT_schedule_flag value.

getFreeCAMode()

```
public boolean getFreeCAMode()
```

Retrieve the free_CA_mode value of this service, false indicates none of the components of this service are scrambled.

Returns:

The free_CA_mode value of this service.

getName()

```
public java.lang.String getName()
```

This method returns the name of the service represented by this service. If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` is one of those in the multilingual_service_name_descriptor, return the name in that language, otherwise return an implementation dependent selection between the names in the multilingual_service_name_descriptor and the name in the service_descriptor. If this descriptor is not present "" is returned. All control characters as defined in ETR 211 [i.3] are ignored. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

Returns:

The name of this service.

getOriginalNetworkID()

```
public int getOriginalNetworkID()
```

Get the original network identification.

Returns:

The original network identification identifier.

getProviderName()

```
public java.lang.String getProviderName()
```

This method returns the service provider name of this service If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` is one of those in the multilingual_service_name_descriptor, return the name in that language, otherwise return an implementation dependent selection between the names in the multilingual_service_name_descriptor and the name in the service_descriptor. If this descriptor is not present "" is returned. All control characters as defined in ETR 211 [i.3] 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 [i.3]) of the service provider of this service without emphasis marks. If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` is one of those in the multilingual_service_name_descriptor, return the name in that language, otherwise return an implementation dependent selection between the names in the multilingual_service_name_descriptor and the name in the service_descriptor. If the descriptor is not present, "" is returned. If the string can be found but does not contain control codes for abbreviating it, the full string shall be returned. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

Returns:

The short service provider name of this service.

getShortServiceName()

```
public java.lang.String getShortServiceName()
```

This method returns the short name (ETR 211 [i.3]) of this service without emphasis marks. If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` is one of those in the multilingual_service_name_descriptor, return the name in that language, otherwise return an implementation dependent selection between the names in the multilingual_service_name_descriptor and the name in the service_descriptor. If the descriptor is not present, "" is returned. If the string can be found but does not contain control codes for abbreviating it, the full string shall be returned. For each character the DVB-SI 8 bit character code is mapped to the appropriate Unicode representation.

Returns:

The short name of this service.

getSIServiceType()

```
public short getSIServiceType()
```

Get the service type. The service type is extracted from the service_descriptor.

Returns:

The service type. (Some of the possible values are defined in the SIServiceType interface.)

See Also:

`SIServiceType`

getTransportStreamID()

```
public int getTransportStreamID()
```

Get the transport stream identification.

Returns:

The transport stream identification identifier.

retrieveFollowingSIEvent(short, Object, SIRetrievalListener, short[])

```
public org.dvb.si.SIRequest retrieveFollowingSIEvent(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve information associated with the following event from the EIT-present/following.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SIEvent interface. If no matching object was found, the appropriate one of the following events is sent: SINotInCacheEvent SIObjectNotInTableEvent or SITableNotFoundEvent.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptor-Tags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid.

See Also:

`SIRequest, SIRetrievalListener, SIEvent, DescriptorTag`

retrievePMTService(short, Object, SIRetrievalListener, short[])

```
public org.dvb.si.SIRequest retrievePMTService(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve the PMTService information associated with this service.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the PMTService interface. If no matching object was found, the appropriate one of the following events is sent: SINotInCacheEvent SIObjectNotInTableEvent or SITableNotFoundEvent.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptor-Tags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid.

See Also:

`SIRequest, SIRetrievalListener, PMTService, DescriptorTag`

retrievePresentSIEvent(short, Object, SIRetrievalListener, short[])

```
public org.dvb.si.SIRequest retrievePresentSIEvent(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

Retrieve information associated with the present event from the EIT-present/following.

The SIIterator that is returned with the event when the request completes successfully will contain an object that implements the SIEvent interface. If no matching object was found, the appropriate one of the following events is sent: SINotInCacheEvent SIObjectNotInTableEvent or SITableNotFoundEvent.

Parameters:

`retrieveMode` - Mode of retrieval indicating whether the data should be retrieved only from the cache (FROM_CACHE_ONLY), from the cache if available and if not from the stream (FROM_CACHE_OR_STREAM), or always from the stream (FROM_STREAM_ONLY).

`appData` - An object supplied by the application. This object will be delivered to the listener when the request completes. The application can use this objects for internal communication purposes. If the application does not need any application data, the parameter can be null.

`listener` - SIRetrievalListener that will receive the event informing about the completion of the request.

`someDescriptorTags` - A list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If someDescriptor-Tags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid.

See Also:

`SIRequest, SIRetrievalListener, SIEvent, DescriptorTag`

retrieveScheduledSIEvents(short, Object, SIRetrievalListener, short[], Date, Date)

```
public org.dvb.si.SIRequest retrieveScheduledSIEvents(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags, java.util.Date startTime,
java.util.Date endTime)
throws SIIllegalArgumentException, SIInvalidPeriodException
```

Retrieve information associated with the scheduled events within the service for a requested period from the EIT-schedule. The events are presented in the order they are present in the EIT-schedule. A scheduled event is retrieved by this method if the time interval from the start time of the event (inclusive) (as returned by SIEvent.getStartTime) to the end time of the event (exclusive) (as defined by the sum of SIEvent.getStartTime and SIEvent.getDuration) intersects the time interval from startTime (inclusive) to endTime (exclusive) specified by the input parameters to this method.

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 someDescriptor-Tags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

`startTime` - The beginning of the required period in UTC time.

`endTime` - The end of the required period in UTC time.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid.

`SIInvalidPeriodException` - When no valid period is indicated.

See Also:

`SIRequest, SIRetrievalListener, SIEvent, DescriptorTag`

org.dvb.si

SIServiceType

Declaration

`public interface` **`SIServiceType`**

Description

This interface defines constants corresponding to the different service types.

See Also:

`SIService.getSIServiceType()`

Fields

D_D2_MAC

`public static final short` **`D_D2_MAC`**

Constant value for the service type as specified in EN 300 468 [4].

DATA_BROADCAST

`public static final short` **`DATA_BROADCAST`**

Constant value for the service type as specified in EN 300 468 [4].

DIGITAL_RADIO_SOUND

`public static final short` **`DIGITAL_RADIO_SOUND`**

Constant value for the service type as specified in EN 300 468 [4].

DIGITAL_TELEVISION

`public static final short` **`DIGITAL_TELEVISION`**

Constant value for the service type as specified in EN 300 468 [4].

FM_RADIO

`public static final short` **`FM_RADIO`**

Constant value for the service type as specified in EN 300 468 [4].

MHP_APPLICATION

`public static final short` **`MHP_APPLICATION`**

Constant value for the service type as specified in EN 300 468 [4].

MOSAIC

`public static final short` **`MOSAIC`**

Constant value for the service type as specified in EN 300 468 [4].

NTSC

`public static final short` **`NTSC`**

Constant value for the service type as specified in EN 300 468 [4].

NVOD_REFERENCE

```
public static final short NVOD_REFERENCE
```

Constant value for the service type as specified in EN 300 468 [4].

NVOD_TIME_SHIFTED

```
public static final short NVOD_TIME_SHIFTED
```

Constant value for the service type as specified in EN 300 468 [4].

PAL

```
public static final short PAL
```

Constant value for the service type as specified in EN 300 468 [4].

SECAM

```
public static final short SECAM
```

Constant value for the service type as specified in EN 300 468 [4].

TELETEXT

```
public static final short TELETEXT
```

Constant value for the service type as specified in EN 300 468 [4].

UNKNOWN

```
public static final short UNKNOWN
```

Constant value for the service type as specified in EN 300 468 [4].

org.dvb.si

SISuccessfulRetrieveEvent

Declaration

```
public class SISuccessfulRetrieveEvent extends SIRetrievalEvent
```

```
java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.si.SIRetrievalEvent
              |
              +--org.dvb.si.SISuccessfulRetrieveEvent
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

This event is sent in response to a SI retrieval request when the retrieve request was successfully completed. The result of the request can be obtained from the getResult method.

See Also:

```
SIRetrievalListener
```

Constructors

SISuccessfulRetrieveEvent(Object, SIRequest, SIIterator)

```
public SISuccessfulRetrieveEvent(java.lang.Object appData, org.dvb.si.SIRequest request,
org.dvb.si.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 org.dvb.si.SIIterator getResult()
```

Returns the requested data in an SIIterator object.

Returns:

An SIIterator containing the requested objects.

See Also:

`SIObjectNotInTableEvent`

org.dvb.si

SITableNotFoundEvent

Declaration

```
public class SITableNotFoundEvent extends SIRetrievalEvent
```

```
java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.si.SIRetrievalEvent
              |
              +--org.dvb.si.SITableNotFoundEvent
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

This event is sent in response to a SI retrieval request when the SI table that should contain the requested information could not be retrieved. The reason may be that the requested table is not broadcast in the transport stream currently associated with the SI database.

See Also:

```
SIRetrievalListener
```

Constructors

SITableNotFoundEvent(Object, SIRequest)

```
public SITableNotFoundEvent(java.lang.Object appData, org.dvb.si.SIRequest request)
```

The constructor for the event.

Parameters:

`appData` - the application data passed in the request method call.

`request` - the SIRequest instance which is the source of the event.

org.dvb.si

SITableUpdatedEvent

Declaration

```
public class SITableUpdatedEvent extends SIRetrievalEvent
```

```
java.lang.Object
  |
  +--java.util.EventObject
        |
        +--org.dvb.si.SIRetrievalEvent
              |
              +--org.dvb.si.SITableUpdatedEvent
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

This event is sent in response to a SI descriptor retrieval request when the table carrying the information about the object has been updated and the set of descriptors consistent with the old object can not be retrieved. The application should in this case first update the SIInformation object and then request the descriptors again.

See Also:

```
SIRetrievalListener
```

Constructors

SITableUpdatedEvent(Object, SIRequest)

```
public SITableUpdatedEvent(java.lang.Object appData, org.dvb.si.SIRequest request)
```

The constructor for the event.

Parameters:

`appData` - the application data passed in the request method call.

`request` - the SIRequest instance which is the source of the event.

org.dvb.si
SITime

Declaration

```
public interface SITime extends SIInformation
```

All Superinterfaces:

```
SIInformation
```

Description

This interface represents the Time and Date Table (TDT) and the (optional) Time Offset Table (TOT). When it represents a TDT table, the `retrieveDescriptors` and `getDescriptorTags` methods behave as documented in the case when there are no descriptors, because the TDT does not contain any descriptors.

See Also:

```
SIDatabase
```

Methods

getUTCTime()

```
public java.util.Date getUTCTime()
```

Get the UTC time as coded in the TDT or TOT table.

Returns:

The UTC as coded in the TDT or TOT table.

org.dvb.si
SITransportStream

Declaration

```
public interface SITransportStream extends SIInformation
```

All Superinterfaces:

```
SIInformation
```

All Known Subinterfaces:

```
SITransportStreamBAT, SITransportStreamNIT
```

Description

This interface is the base interface for representing information about transport streams.

Transport stream retrieval methods in the SIDatabase class and the SINetwork interface use the NIT table and will return objects that implement the SITransportStreamNIT interface.

Transport stream retrieval methods in the SIBouquet interface use the BAT table and will return objects that implement the SITransportStreamBAT interface.

Methods

getDvbLocator()

```
public org.davic.net.dvb.DvbLocator getDvbLocator()
```

Gets a DvbLocator that identifies this transport stream.

Returns:

The DvbLocator of this transport stream.

getOriginalNetworkID()

```
public int getOriginalNetworkID()
```

Get the original network identification.

Returns:

The original network identification identifier.

getTransportStreamID()

```
public int getTransportStreamID()
```

Get the transport stream identification.

Returns:

The transport stream identification identifier.

retrieveSIServices(short, Object, SIRetrievalListener, short[])

```
public org.dvb.si.SIRequest retrieveSIServices(short retrieveMode, java.lang.Object appData,
org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags)
throws SIIllegalArgumentException
```

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 someDescriptor-Tags is null, the application is not interested in descriptors. All values that are out of the valid range for descriptor tags (i.e. 0...255) are ignored, except for the special meaning of -1 as the only element in the array.

Returns:

An SIRequest object.

Throws:

`SIIllegalArgumentException` - thrown if the retrieveMode is invalid.

See Also:

SIRequest, SIRetrievalListener, SIService, DescriptorTag.

org.dvb.si
SITransportStreamBAT

Declaration

```
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

Declaration

```
public interface SITransportStreamDescription extends SIInformation
```

All Superinterfaces:

SIInformation

Description

This interface represents the Transport Stream Description Table (TSDT).

It defines no methods of its own other than those inherited from SIInformation.

See Also:

SIDatabase, SITransportStream

org.dvb.si

SITransportStreamNIT

Declaration

```
public interface SITransportStreamNIT extends SITransportStream
```

All Superinterfaces:

```
SIInformation, SITransportStream
```

Description

This interface represents information about transport streams that has been retrieved from a NIT table. All descriptor accessing methods return descriptors retrieved from a NIT table. Methods in SIDatabase and SINetwork for retrieving transport streams return objects that implement this interface.

Methods

getNetworkID()

```
public int getNetworkID()
```

Get the identification of the network this transport stream is part of.

Returns:

The network identification identifier.

org.dvb.si

SIUtil

Declaration

```
public class SIUtil

java.lang.Object
  |
  +--org.dvb.si.SIUtil
```

Description

This class contains SI related utility functions.

Methods

convertSIStringToJavaString(byte[], int, int, boolean)

```
public static java.lang.String convertSIStringToJavaString(byte[] dvbSIText, int offset, int length,
boolean emphasizedPartOnly)
throws SIIllegalArgumentException
```

This method converts a text string that is coded according to annex A of the DVB-SI specification (EN 300 468) to a Java String object.

The text that must be converted is contained in "dvbSIText" from index "offset" to index "offset+length-1" (inclusive).

If the text that must be converted is not validly coded according to annex A of the DVB-SI specification, then the result is undefined.

Parameters:

`dvbSIText` - The byte array that contains the string that must be converted.

`offset` - The offset indicates the start of the DVB-SI text in dvbSIText.

`length` - Length of the DVB-SI text in bytes.

`emphasizedPartOnly` - If emphasizedPartOnly is true, then only the text that is marked as emphasized (using the character emphasis on [0x86] and character emphasis off [0x87] control codes) will be returned. Otherwise, the character emphasis codes will be ignored, and all of the converted text will be returned.

Returns:

The converted text.

Throws:

`SIIllegalArgumentException` - thrown if offset and/or offset+length-1 is not a valid index in dvbSIText.

org.dvb.si

TextualServiceIdentifierQuery

Declaration

```
public interface TextualServiceIdentifierQuery
```

All Known Subinterfaces:

`SIService`

Description

An interface that can be implemented by objects representing DVB services. Allows applications to obtain the textual service identifiers related to a service.

Since:

MHP1.0.1

Methods

getTextualServiceIdentifiers()

```
public java.lang.String[] getTextualServiceIdentifiers()
```

Returns the textual service identifiers related to this object.

Returns:

An array of String objects containing the textual service identifiers or null if none are present.

Since:

MHP1.0.1.

Annex N (normative): Streamed Media API Extensions

GEM [1], annex N is included in the present document, with the following notes and modifications.

For MHP, signalling for the active format description shall be supported as defined in annex B of TS 101 154 [i.1] (this is optional in GEM).

Annex O (normative): Integration of the Java TV SI API and DVB SI

O.1 Introduction

This clause describes how the Java TV Service Information API as described in Java TV [22] can be mapped to the data structures of DVB Service Information as defined in EN 300 468 [4]. Secondly the present document describes how the Java TV API and the DVB SI API (as described in annex M "(normative): SI Access API") can be integrated.

O.2 Mapping of the Java TV SI API to DVB SI

This clause describes for every relevant Java interface and method in the Java TV SI API how it is mapped to the DVB Service Information.

When adding a listener to monitor for changes in an SI table (or data carried in an SI table), an event shall not be generated for the current version of that table (or data) found in the network at the time the listener is added. Events shall only be generated for changes following the detection of that current version.

O.2.1 javax.tv.service.Service

MHP terminals normally maintain a stored list of "installed" services discovered through the process explained in clause Z.6 "How does the platform know which services are available?". The Service interface shall represent the entries in this list and also those services which are discovered in the network but which are not yet stored in this list (see clause O.2.8.5, "getService"). Neither PSI-only services nor hidden services should be presented to the end-user by the navigator, either in a UI showing the list of available services or in response to the program up / program down keys on conventional remote controls.

Depending on the MHP terminal implementation, the objects implementing this interface may or may not implement the ServiceNumber interface as well. Furthermore, it is allowed that even within the same MHP terminal implementation, some objects implementing the Service interface implement the ServiceNumber while other objects implement only the Service interface.

Objects implementing this interfaces and representing DVB services shall also implement the `org.dvb.si.TextualServiceIdentifierQuery` interface.

The methods are mapped as follows.

O.2.1.1 getName

Returns the name of the service as stored in the MHP terminal. Depending on the MHP terminal implementation, the end user may have the possibility to edit these names according to his preferences. If the contents of this field are retrieved by the MHP terminal by default from DVB SI, it is recommended that the MHP terminal uses the abbreviated form of the service name from the Service descriptor (see clause 4.6.1, "Use of control codes in names" in ETR 211 [i.3]).

O.2.1.2 getServiceType

Returns the ServiceType according to the mapping defined in clause O.2.3 "javax.tv.service.ServiceType" on page 763.

O.2.2 javax.tv.service.navigation.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 getName

If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` corresponds to the language of a multilingual component descriptor or data broadcast descriptor in the EIT, return the description in that language. Otherwise return an implementation dependent selection between the descriptions available in the EIT.

O.2.2.2 getAssociatedLanguage

Returns the ISO 639-2 [13] language code indicating the language of the component (i.e. not necessarily the selected language for the name returned by `getName()`) from the component descriptor, multilingual component descriptor or data broadcast descriptor.

O.2.2.3 getStreamType

Returns the stream type according to the mapping from the stream_content field and the component_type field of the Component descriptor or the Data broadcast descriptor to the Java TV stream types according to clause O.2.4 "javax.tv.service.navigation.StreamType".

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 Java TV service types as follows.

Table O.1: Mapping DVB to Java TV service types

DVB Service type code	DVB Service Type Description	Java TV 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 to 0x0F, 0x11 to 0xFF		UNKNOWN

O.2.4 javax.tv.service.navigation.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 Java TV 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 and component types to Java TV

DVB stream_content	DVB component_type	Java TV Stream type
0x01	0x00 to 0xff	VIDEO
0x02	0x00 to 0xff	AUDIO
0x03	0x01, 0x10 to 0x13, 0x20 to 0x23	SUBTITLES
0x03	0x02	DATA
0x03	0x00, 0x14 to 0x1F, 0x24 to 0xFF	UNKNOWN
0x04 to 0x0F	0x00 to 0xFF	UNKNOWN

O.2.5 javax.tv.service.SIElement

This interface is implemented by objects implementing the Network, Bouquet, TransportStream, ServiceDetails, ServiceComponent and ProgramEvent interfaces.

O.2.5.1 getServiceInformationType

This method shall return the DVB_SI ServiceInformationType.

O.2.6 javax.tv.service.SIManager

O.2.6.1 getSupportedDimensions

The parental rating descriptor defined in DVB SI standardizes one rating scheme that is based on age. To describe this DVB defined rating scheme, the getSupportedDimensions shall return an array that contains the string "DVB Age based rating".

O.2.6.2 getRatingDimension

When given the string "DVB Age based rating", this method shall return an object implementing the RatingDimension interface as described in clause O.2.10 "javax.tv.service.RatingDimension".

O.2.6.3 retrieveSIElement

When passed a locator that points to a service, an object implementing the ServiceDetails interface shall be returned. Locators representing program events are not supported and shall fail with an `SIRequestFailureType(INSUFFICIENT_RESOURCES)`.

Other types of locators are supported as defined.

NOTE: program events can still be retrieved for specific times using the methods in `javax.tv.service.guide.ProgramSchedule`.

O.2.6.4 getTransports

The object returned by this method shall implement the Transport interface as described in clause O.2.12 "javax.tv.service.transport.Transport".

O.2.6.5 filterServices

Filtering of Services shall be supported with ServiceFilters. The SIElementFilter is required to be supported as defined in clause O.2.7, "javax.tv.service.navigation.SIElementFilter".

O.2.6.6 retrieveProgramEvent

This method shall fail with a `SIRequestFailureType(INSUFFICIENT_RESOURCES)`.

NOTE: program events can still be retrieved for specific times using the methods in `javax.tv.service.guide.ProgramSchedule`.

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 clause O.2.9, "javax.tv.service.navigation.CAIdentification". These objects shall not implement the ServiceNumber interface.

O.2.8.1 getLongName

If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` corresponds to the language of a multilingual service descriptor in the SDT, return the service name in that language. Otherwise return an implementation dependent selection between the descriptors available in the SDT.

O.2.8.2 getServiceType

Returns the ServiceType according to the mapping defined in clause O.2.3, "javax.tv.service.ServiceType".

O.2.8.3 retrieveServiceDescription

Shall always result in a notifyFailure of the SIRequestor object being called with the DATA_UNAVAILABLE SIRequestFailureType, as DVB SI does not include a service description.

O.2.8.4 retrieveComponents

If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` corresponds to the language of a multilingual component descriptors or data broadcast descriptors in the EIT present/following table then the information for the ServiceComponents shall be retrieved from those descriptors. Otherwise the information for the ServiceComponents shall be returned from an implementation dependent selection between the descriptors available in the EIT present/following table.

O.2.8.5 getService

Calling this method for a `ServiceDetails` instance whose corresponding service is not in the "installed" services list (i.e. a new service found in the SDT which would not previously have been returned from `SIManager.FilterServices(null)`) shall return a Service instance and shall not return null. The implementation is responsible for creating this Service instance. It is implementation dependent whether this Service instance is also "installed" and hence whether or not it is returned by `FilterServices(null)`. This Service instance shall have no different behaviour from "installed" services apart from this.

O.2.9 javax.tv.service.navigation.CAIdentification

This interface shall be implemented by objects implementing the ServiceDetails, ProgramEvent or Bouquet interface.

O.2.9.1 getCASystemIds

Returns the array of integer values containing the CA_system_ids from the CA identifier descriptor. If the CA identifier descriptor is not present, returns an empty array.

O.2.9.2 isFree

When implemented in an object implementing the ServiceDetails or ProgramEvent interface, this method shall return true if and only if the free_CA_mode bit is set to "0" in the SDT or EIT entry, respectively.

When implemented in an object implementing the Bouquet interface, this method shall return true if and only if there is no CA identifier descriptor present in the BAT.

O.2.10 javax.tv.service.RatingDimension

The Parental rating descriptor defined in DVB SI standardizes one rating scheme that is based on age. This rating scheme contains 15 distinct age rating levels from 4 years to 18 years.

An object that describes this DVB defined rating scheme shall implement the methods as follows.

O.2.10.1 getDimensionName

Returns the string "DVB Age based rating".

O.2.10.2 getNumberOfLevels

Returns 15.

O.2.10.3 getRatingLevelDescription

Returns an array of 2 strings of the form:

{"Over *n*", "Recommended minimum age: *n* years"}

where n is the input parameter + 4.

O.2.11 javax.tv.service.navigation.ServiceProviderInformation

This interface shall be implemented by objects implementing the ServiceDetails interface.

O.2.11.1 getProviderName

If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` corresponds to the language of a multilingual service descriptor or service descriptor in the SDT, return the provider name from that descriptor. Otherwise return an implementation dependent selection from the descriptors available in the SDT.

O.2.12 javax.tv.service.transport.Transport

The object implementing the Transport interface shall also implement the interfaces NetworkCollection and BouquetCollection.

O.2.13 javax.tv.service.transport.Bouquet

The Bouquet interface is implemented by an object that represents a DVB SI Bouquet.

Objects implementing this interface shall also implement the CAIdentification interface. See clause O.2.9, "javax.tv.service.navigation.CAIdentification".

O.2.13.1 getBouquetID

Returns the integer DVB SI Bouquet ID value.

O.2.13.2 getName

If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` corresponds to the language of a multilingual bouquet name descriptor in the BAT, return the name in that language. Otherwise return an implementation dependent selection from the descriptors available in the BAT.

O.2.13.3 getLocator

Returns an implementation dependent javax.tv.locator.Locator object that does not have a standardized external representation and might not be a org.davic.net.dvb.DvbLocator.

O.2.14 javax.tv.service.transport.Network

The Network interface is implemented by an object that represents a DVB SI Network.

O.2.14.1 getNetworkID

Returns the integer DVB SI Network ID value.

O.2.14.2 getName

If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` corresponds to the language of a multilingual network name descriptor in the NIT, return the name in that language. Otherwise return an implementation dependent selection from the descriptors available in the NIT.

O.2.14.3 getLocator

Returns an implementation dependent javax.tv.locator.Locator object that does not have a standardized external representation and might not be a org.davic.net.dvb.DvbLocator.

O.2.15 javax.tv.service.transport.TransportStream

The TransportStream interface is implemented by an object that represents a transport stream.

O.2.15.1 getTransportStreamID

Returns the integer DVB SI transport stream ID value.

O.2.15.2 getDescription

Transport streams do not have descriptions in DVB SI, so this method shall return an empty string.

O.2.16 javax.tv.service.guide.ProgramEvent

This interface is implemented by objects representing DVB SI Events.

Objects implementing this interface shall also implement the CAIdentification interface. See clause O.2.9, "javax.tv.service.navigation.CAIdentification".

O.2.16.1 getDuration

Returns the duration value from the event entry in the body of the EIT.

O.2.16.2 getStartTime

Returns the start time value from the event entry in the body of the EIT.

O.2.16.3 getEndTime

Returns the end time value calculated from the start time and duration in the body of the EIT.

O.2.16.4 getName

If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` corresponds to the language of a short event descriptor in the EIT, return the name from that descriptor. Otherwise return an implementation dependent selection from the descriptors available in the EIT.

O.2.16.5 retrieveDescription

If the language returned by `javax.tv.service.SIManager.getPreferredLanguage` corresponds to the language of a short event descriptor in the EIT, return the description from that descriptor. Otherwise return an implementation dependent selection from the descriptors available in the EIT.

The description text in the `ProgramEventDescription` object is just passed through as a String containing the description as it was transmitted in the EIT table with just a character set mapping performed.

Codes 0x8a and 0xe08a defined in tables A.1 and A.2 of EN 300 468 [4] shall be mapped to the Java newline character, '\n'.

O.2.16.6 getRating

Returns the rating from the Parental rating descriptor according to the mapping defined in clause O.2.17 "javax.tv.service.guide.ContentRatingAdvisory".

O.2.17 javax.tv.service.guide.ContentRatingAdvisory

O.2.17.1 getDimensionNames

Returns an array containing the string "DVB Age based rating" as one of the elements in the array.

O.2.17.2 getRatingLevel

When the parameter is "DVB Age based rating" and the parental rating descriptor contains a rating value between 0x01 and 0x0F for the current region, returns the integer rating value contained in the parental rating descriptor decremented by one (i.e. the value in the descriptor - 1). In other cases when the parameter is "DVB Age based rating" returns -1.

O.2.17.3 getRatingText

When the parameter is "DVB Age based rating" and the parental rating descriptor contains a rating value between 0x01 and 0x0F for the current region, returns a string of the form "Recommended minimum age: n years" where n is the rating value in the descriptor incremented by 3 (i.e. the value in the descriptor + 3). In other cases when the parameter is "DVB Age based rating" returns an empty string.

NOTE: Applications wishing to present this information in languages other than English should use the `getRatingLevel()` method and perform their own encoding of this for end-user presentation.

O.2.17.4 getDisplayText

When the parental rating descriptor contains a rating value between 0x01 and 0x0F for the current region, returns a string that contains the string "Over n", where n is the rating value in the descriptor incremented by 3 (i.e. the value in the descriptor + 3), as its substring.

NOTE: Applications wishing to present this information in languages other than English should use the `getRatingLevel()` method and perform their own encoding of this for end-user presentation.

O.2.17.5 retrieveProgramEvent(Locator, SIRequestor)

This method is not supported and shall fail with an `SIRequestFailureType(INSUFFICIENT_RESOURCES)`.

O.3 Integration of the Java TV 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 Java TV SI API interfaces. When retrieving SI objects through the Java TV 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

Figure O.1

Annex P (normative): Broadcast Transport Protocol Access

P.1 Overview

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: The protocol is abstracted from the application viewpoint, so, objects accessible through this API are either objects encapsulated in an Object Carousel, or Objects located in an interactive DSMCC network on a remote server.

To benefit from the fact that most of the functionalities are already covered by the java.io package, this API inherits from java.io and only defines the extra-functionalities pertaining to:

a) The nature of the network (broadcast or DSMCC remote server) and its latency (e.g. possibility to asynchronously load the objects).

b) The type of the objects that can be encapsulated in a carousel and that do not exist in a classical File structure. These are: ServiceGateway, Directory, File, Stream and StreamEvent.

c) Definition of ServiceGateway, which defines a new namespace corresponding to the new Domain, and enables the mounting of a new volume.

An application can optionally use only the classes of java.io. Alternatively/additionally applications can use additional classes and methods adapted to the specific nature and latency of the network (such as for example, the asynchronous loading of objects).

The following, briefly explains the functionalities offered by this API.

The ServiceDomain class enables attaching to a ServiceDomain. Attachment to a ServiceDomain corresponds to the mounting of a volume in the file hierarchy system and the loading of the Service Gateway.

When attached to a Service Domain the DSM-CC UU-File, UU-Stream, UU-Directory and UU-StreamEvent objects are accessible through this API.

The class DSMCCObject represents a UU-object. Due to the close relationship between resident files and downloaded files, this class inherits from the java.io.File class. The DSMCCObject class just defines the additional methods specific to DSMCC-UU that basically deal with asynchronous or synchronous loading of Objects.

For the UU-Files or UU-Directory Objects, their content is accessible as it would be for a classical file system, i.e. by using the `java.io package` (e.g. for listing the objects pointed to by a Directory object, you invoke the `list()` method of the `java.io.File` class, or to access the content of a UU-File, you can instantiate a `FileInputStream` to read the `File`, etc.).

Additionally, the DSMCCStream and DSMCCStreamEvent classes define functionalities specific to the respective types of Objects (Stream and StreamEvent), which basically consists in accessing the attributes of these Objects.

The DSMCCStream class provides access to the following attributes Duration, current NPT. In addition, an application can retrieve the list of Taps (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 AsynchronousLoadingEvent class and its subclasses represent events that are sent to a listener to notify it of the loading of an Object that had been activated by the application (asynchronous loading mode).

The StreamEvent class represents an abstraction of the real event that is generated, i.e. the streameventdescriptor, which enables the broadcaster to synchronize the application with the stream. This class enables the access to the content of an event, the content of the event being described by the StreamEventDescriptor, which is inserted in the stream in DSMCC sections at the transport level.

Finally, the StreamEventListener and AsynchronousLoadingEventListener are interfaces that must be implemented by the application, in order for it to receive the respective StreamEvents and AsynchronousLoadingEvents.

P.2 Javadoc for org.dvb.dsmcc package

GEM [1], clause P.4 is included in the present document.

Annex Q (normative): Datagram Socket Buffer Control

GEM [1], annex Q is included in the present document.

Annex R (normative): DVB-J Return Channel Connection Management API

GEM [1], annex R is included in the present document.

Annex S (normative): Application Listing and Launching

GEM [1], annex S is included in the present document, with the following notes and modifications.

S.1 Limitations on database filter types

An MHP terminal can define only two types of application database filters: `CurrentServiceFilter` and `RunningApplicationsFilter`. For MHP, therefore, the methods `org.dvb.application.getAppIDs(AppDatabaseFilter)` and `org.dvb.application.getAppAttributes(AppsDatabaseFilter)` on the class `AppsDatabase` shall contain the following text:

- For implementations conforming to this version of the specification, only `CurrentServiceFilter` or `RunningApplicationsFilter` filters may return a non-empty `Enumeration`. If the filter object is not an instance of `CurrentServiceFilter` or `RunningApplicationsFilter` or a subclass of either then, the method shall return an empty `Enumeration`.

S.2 AppProxy Interface

The note accompanying the description of the DESTROYED field regarding legacy behavior for packaged media applications is not included in the present document.

S.3 DVBHTMLProxy Interface

The present document includes an additional interface, `org.dvb.application.DVBHTMLProxy`:

org.dvb.application

Interface DVBHTMLProxy

Declaration:

```
public interface DVBHTMLProxy
extends AppProxy
```

A `DVBHTMLProxy` Object is a proxy to a DVBHTML application.

Fields

LOADING

```
static final int LOADING
```

The application is in the loading state.

Since:

MHP 1.0.2.

KILLED

```
static final int KILLED
```

The application is in the killed state.

Since:

MHP 1.0.2.

Methods

prefetch

```
void prefetch()
```

Loads the initial entry page of the application and waits for a signal. This method mimics the PREFETCH control code and is intended to be called instead of and not as well as start. Calling prefetch on a started application will have no effect.

Throws:

`java.lang.SecurityException` - if the calling application does not have permission to start applications.

Since:

MHP1.0.

startTrigger

```
void startTrigger(java.util.Date starttime)
```

Sends the application a start trigger at the specified time.

Parameters:

`starttime` - the specified time to send a start trigger to the application. If the time has already passed the application manager shall send the trigger immediately. Dates pre-epoch shall always cause the application manager to send the trigger immediately.

Throws:

`java.lang.SecurityException` - if the calling application does not have permission to start applications.

Since:

MHP1.0.

trigger

```
void trigger(java.util.Date time,
             java.lang.Object triggerPayload)
```

Sends the application a trigger with the given payload at the specified time.

Parameters:

`time` - the specified time to send a start trigger to the application. If the time has already passed the application manager should send the trigger immediately. Dates pre-epoch shall always cause the application manager to send a 'now' trigger.

`triggerPayload` - the specified payload to deliver with the trigger. The payload is specified as object, but this will be refined once DVB-HTML Triggers are properly defined.

Throws:

`java.lang.SecurityException` - if the calling application does not have permission to start applications.

Since:

MHP1.0.

Annex T (normative): Permissions

GEM [1], annex T is included in the present document.

Annex U (normative): Extended graphics APIs

GEM [1], annex U is included in the present document.

Annex V:
Void

Annex W (informative): DVB-J examples

W.1 DVB-J Application lifecycle implementation example

```
import org.havi.ui.HScene;
import org.havi.ui.HSceneFactory;
import java.awt.Component;
import java.awt.BorderLayout;
import org.dvb.ui.DVBColor;
import javax.tv.xlet.Xlet;
import javax.tv.xlet.XletContext;
import javax.tv.xlet.XletStateChangeException;


public class HelloDVB extends Component implements Xlet {

    private XletContext context;
    private HScene scene;

    public void initXlet(XletContext context) throws XletStateChangeException {
        this.context = context;
        scene = HSceneFactory.getInstance().getDefaultHScene();
        scene.setBounds( 90,72,540,432 );
        scene.setLayout(new BorderLayout(0, 0));
        scene.add(this, "Center");
    }

    public void startXlet() throws XletStateChangeException {
        scene.setVisible(true);
        requestFocus();
    }

    public void pauseXlet() {
        scene.setVisible(false);
    }

    public void destroyXlet(boolean unconditional) throws XletStateChangeException {
        scene.dispose();
    }

    public void paint(java.awt.Graphics g) {
        g.setColor(new DVBColor(0, 0, 70, 180));
        g.fillRect(0, 0, getSize().width, getSize().height);
        g.setColor(DVBColor.yellow);
        g.drawString("Hello DVB", (getSize().width-110) / 2, getSize().height / 2);
    }
}
```

A simple example of Xlet lifecycle is a stock ticker application that uses a back channel to retrieve stock quotes, which it displays on the viewer's television.

a) The application manager retrieves the Xlet's code.

b) The application manager creates an instance of the XletContext Object and initializes it for the new Xlet.

c) The application manager initializes the Xlet by calling its `initXlet()` method and passing it the context object.

d) The Xlet uses the context object to initialize itself and enters the `Paused` state.

e) The application manager calls the Xlet's `startXlet()` method. The application manager assumes that the Xlet is performing its service.

f) Upon receiving this signal, the Xlet creates a new thread that opens the back channel to retrieve the stock quotes. The Xlet is now in the `Active` state.

g) The Xlet begins to show the stock quotes.

h) Due to circumstances beyond the control of the Xlet, it is no longer able to retrieve updated stock quotes.

i) The Xlet decides to continue displaying the most recent quotes it has.

NOTE: The Xlet is still in the `Active` state.

j) After a time, the Xlet is still unable to open the back channel. It decides that the quotes it is displaying are too old to present and that it can no longer perform its service. It chooses to take itself out of the `Active` state. It calls the `paused()` method on XletContext to signal this change to the application manager.

k) Finally, the Xlet decides it no longer has any chance of performing its service, so it decides it should be terminated. It calls the `destroyed()` method on the XletContext to signal application manager that it has entered the `Destroyed` state. The Xlet does some final clean up.

l) The application manager prepares the Xlet for garbage collection.

W.2 Example of exporting an object for inter-application communication

```
public interface MyService extends java.rmi.Remote {
        public String getData() throws java.rmi.RemoteException;
    }

    public class MyServer implements MyService {

        private static javax.tv.xlet.XletContext ctx;

        public String getData() throws java.rmi.RemoteException {
            return "Hello from " + ctx.getXletProperty("dvb.app.id");
        }

        //
        // Called upon Xlet initialization
        //
        public static void export(XletContext ctx) {
            this.ctx = ctx;
            Remote server = new MyServer();
            org.dvb.io.ixc.IxcRegistry.bind(ctx, "myserver", server);
        }

        //
        //  Try to import the object that we previously exported.
        // Note that this would typically be done from a different
        // Xlet, but importing from yourself works, too.
        // Called when the Xlet is run.
        //
        public static void import(XletContext ctx) {
            String appId = (String) ctx.getXletProperty("dvb.app.id");
            String orgId = (String) ctx.getXletProperty("dvb.org.id");

            Remote obj;
            try {
                org.dvb.io.ixc.IxcRegistry.lookup(ctx, "/" + orgId + "/" + appId + "/myserver");
                MyService r = (MyService) obj;
                System.out.println("Success " + r
                                            + ", " + r.getData());
            } catch (Exception ex) {
                }
        }
    }
```

W.3 Example of use of video drip feed

```
import java.lang.*;
import java.io.*;
import javax.tv.xlet.*;
import javax.media.*;
import java.net.URL;
```

```
import org.dvb.media.DripFeedPermission;
import org.dvb.media.DripFeedDataSource;

/**
 * VideoDripTest creates an instance of DripFeedDataSource and creates a player
 * using this source.
 */
public class SingleVideoDripTest implements javax.tv.xlet.Xlet
{
    static final int                 MAX_DRIP_DATA_SIZE = 32000;
    private DripFeedDataSource      dripDataSource = null;
    private Player                  dripPlayer = null, oldPlayer = null;

    public void initXlet(XletContext ctx) throws XletStateChangeException
    {

        /* Check if this application has permission to play video drip
         * feed or not.
         */
        System.err.println("initXlet called");

        SecurityManager sm = System.getSecurityManager();

        System.err.println("Checking the permission");

        if (sm != null) {
            try {
                sm.checkPermission(new DripFeedPermission(""));
            }
            catch(java.lang.SecurityException ex) {
                throw new XletStateChangeException(ex.getMessage());
            }
        }

        System.err.println("Exiting initXlet method");
    }

    public void startXlet() throws XletStateChangeException
    {
        byte[] dripData = new byte[MAX_DRIP_DATA_SIZE];

        /* Assumption: No media player is active. If this is not true, we
         * need to get the current player and stop/close it.
         */
        System.err.println("startXlet called");

        try {
            /* Create a data source for video drip */
            dripDataSource = new DripFeedDataSource();

            System.err.println("Got the DataSource");

            /* Create a player and start it */
            dripPlayer = Manager.createPlayer(dripDataSource);
        }
        catch (IOException ioe1)
        {
            System.err.println(ioe1.getMessage());
            throw new XletStateChangeException(ioe1.getMessage());
        }
        catch (NoPlayerException pse)
        {
            System.err.println(pse.getMessage());
            throw new XletStateChangeException(pse.getMessage());
        }

        System.err.println("Starting Drip Player");

        dripPlayer.start();

        System.err.println("Started the player");

        /* Read drip data from a file */
        try
        {
            FileInputStream fin = new FileInputStream("images.mpg");
            fin.read(dripData);
            fin.close();
```

```
        }
        catch (IOException ioe2)
        {
            System.err.println("IOException: " + ioe2.getMessage());
            throw new XletStateChangeException(ioe2.getMessage());
        }
        catch (SecurityException se)
        {
            System.err.println(se.getMessage());
            throw new XletStateChangeException("Security Exception" + se.getMessage());
        }

        System.err.println("Feeding data to the datasource");

        /* Feed the data to the data source */
        dripDataSource.feed(dripData);

        System.err.println("Fed data to the datasource");
    }

    public void pauseXlet()
    {
    }

    public void destroyXlet(boolean unconditional)
    {
        dripPlayer.close();
    }
}
```

W.4 Example of CPU bound animation

```
/**
 * This is an example of doing CPU-bound animation.  This
 * code attempts to do animation as fast as possible, using
 * a low-priority thread so that the animation does not interfere
 * with more important tasks, like responding to user input.  Animation
 * is limited to 25 frames per second, just in case we're running on
 * a really fast box.
 **/

import java.awt.Graphics;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Toolkit;
import org.dvb.ui.DVBBufferedImage;

public abstract class AnimatedView extends Component implements Runnable {

    private Thread worker = null;
    private Graphics theScreen = null;
    private DVBBufferedImage buffer = null;
    private boolean stopping = false;

    /**
     * Called by the Xlet to start animation.  Never to be called
     * when animation is in progress!
     */
    public synchronized void startAnimation() {
        if (worker != null) {
            throw new IllegalStateException();    // It's a bug
        }
        worker = new Thread(this);
        worker.setPriority(2);        // Animation is CPU-bound
        theScreen = getGraphics();    // Component.getGraphics()
        Dimension d = getSize();
        buffer = new DVBBufferedImage(d.width, d.height);
        stopping = false;
        worker.start();
    }

    /**
     * Stop animation, and don't return until it has stopped.
     */
    public synchronized void stopAnimation() {
```

```
    if (worker == null) {
        throw new IllegalStateException();    // it's a bug
    }
    stopping = true;
    for (;;) {        // Wait until stopped
        if (Thread.interrupted()) {        // Xlet being terminated
            return;
        }
        notifyAll();
        try {
            wait();
        } catch (InterruptedException ex) {
           Thread.currentThread().interrupt();
        }
        if (!stopping) {
            theScreen.dispose();
            theScreen = null;
            buffer.flush();
            buffer = null;
            worker = null;
            return;
        }
    }
}

/**
 * Run the animation until stopped.  Called from the worker thread
 * only.  This will probably be CPU-bound, as it sets an ambitious
 * target of 25fps animation.
 */
public void run() {

    long start = System.currentTimeMillis();
    long lastFrameTime = start - 40;
    long now = 0;

    animation:
    for (;;) {

        // Wait until at least 1/25 of a second after last frame, and
        // bail out of thread if we're stopping
        synchronized(this) {
        for (;;) {
            if (stopping) {
                stopping = false;
                notifyAll();
                return;
            }
            if (Thread.interrupted()) {        // Xlet being terminated
                return;
            }
            now = System.currentTimeMillis();
            long delta = now - lastFrameTime;
            if (delta >= 40) {
                break;
            } else {
                try {
                    wait(40 - delta);
                } catch (InterruptedException ex) {
                    Thread.currentThread().interrupt();
                }
            }
        }
    }
    lastFrameTime = now;
    long timeSinceStart = now - start;

    Graphics g = buffer.getGraphics();

    for (int part = 0; part < getNumParts(); part++) {
        drawPart(part, g, timeSinceStart);

        if (shouldStop()) {
            continue animation;    // Bail out if animation should stop
        }
    }

    g.dispose();
```

```
        theScreen.drawImage(buffer, 0, 0, null);
        Toolkit.getDefaultToolkit().sync();

        // It is essential to yield this thread.  In the
        // Java threading model, a CPU-bound low priority
        // thread can prevent a higher priority thread from
        // running if the low priority thread never yields
        // or waits.
        Thread.currentThread().yield();
        }
    }

    private synchronized boolean shouldStop() {
        return Thread.interrupted() || stopping;
    }

    public synchronized void paint(Graphics g) {
        if (worker == null || stopping) {
            // Paint whatever we paint when we're not animating
        } else {
            // Probably do nothing.  If our animation target were
            // significantly less than 25 fps, we'd want to set a variable
            // to cause a repaint ASAP, then call notifyAll() to break
            // the animation loop out of any wait() it might be in.
        }
    }

    /**
     * Draw a part of the animation. For each frame, AnimatedView will call
     * this method first for part 0, then part 1, up to getNumParts()-1.
     * Between each part, AnimatedView will check if animation needs to
     * stop for some reason.
     * <p>
     * Subclasses should ensure that drawing the entire scene is divided
     * into enough parts to ensure that animation can be stopped quickly.
     *
     * @see #getNumParts
     */
    protected abstract void drawPart(int num, Graphics g, long timeSinceStart);

    /**
     * @return the number of parts in this animation.  This determines how
     * many times drawPart will be called.
     *
     * @see #drawPart
     */
    protected abstract int getNumParts();
}
```

W.5 Example of using optional APIs

```
/**
 * This is an example of writing application code that uses
 * an optional API that may not be present on all MHP
 * platforms.  In this case, the application tests for
 * the presence of the class org.dvb.internet.WWWBrowserService.
 * If it is present, then the application has available to
 * it that API, and other APIs needed for internet access.
 * <p>
 * This is the main Xlet class
 **/

public class MyXlet implements javax.tv.xlet.Xlet {
    ...

    public wFeatures wFeatures = null;
    // This object is the gateway to all of the Xlet
    // code that relies on the internet access profile.  If
    // null, the Xlet is running on a box not supporting
    // the internet access profile.

    public void initXlet(javax.tv.xlet.XletContext ctx) {
        ...

        try {
```

```
            Class c = Class.forName(
                "org.dvb.internet.WWWBrowserService");
                    // We have a www browser!
                c = Class.forName("wFeaturesImpl");
                    // An Xlet class
                wFeatures = c.newInstance();
                    // Calls default constructor
        } catch (ClassNotFoundException ex) {
            // No internet profile support, so we leave wFeatures null.
        }

    }
}

/**
 * This interface is used by the Xlet to call into
 * code that uses a www browser in any way.  It
 * is essential that the only code in the Xlet that
 * uses classes not present on the IA profile be reached
 * via the class that implements this interface.  If an
 * Xlet directly references an API from some other place, then
 * the entire Xlet might fail to load on valid MHP
 * implementations.
 * <p>
 * In technical terms, when a class is loaded, a valid Java
 * implementation may load the transitive closure of all
 * statically-referenced classes, and fail to load the first
 * class if any of the other classes are not found.  Thus,
 * if an Xlet directly references an API that is not present,
 * the entire Xlet could fail to load on valid MHP
 * implementations.  As examples, static references can
 * be the types of data members or local variables in code
 * blocks, even if those code blocks are never executed.
 * <p>
 * To put the APIs that might not be present outside of the
 * transitive closure of statically referenced classes, we
 * introduce this interface, and dynamically load the single
 * class that implements it using Class.forName().  The class
 * that implements this interface can contain static references
 * to APIs that might not be present, and it can contain static
 * references to classes that reference such classes.
 *  <p>
 *  This interface can be thought of as a Facade.
 *  See the Facade pattern on page 185 of GoF
 *  ("Design Patterns" by Gamma et al, ISBN
 *  0-201-63361-2).
 **/

public interface wFeatures {

    /**
     * Set up the www browser, and get ready for the Xlet
     * to run.
     **/
    public void init();

    ... Here, there are declarations of all of the features
    of the Xlet that use the www browser.
    There might be other methods, to manage
    the www browser during Xlet state transitions,
    such as to the paused or destroyed state  ...

}


/**
 * This class contains implementations of all features of the
 * xlet that depend on the presence of a www browser.  It
 * can safely reference classes that are not present in the
 * IA profile, and it can safely reference classes that
 * reference such classes.
 **/

public class wFeaturesImpl implements wFeatures {

    private org.dvb.internet.WWWBrowserService;

    public void init() {
```

```
        javax.tv.service.ServiceType s=org.dvb.internet.InternetServiceType.INTERNET_CLIENT;
    }

    ...  Here, there are implementations of the features
    declared above ...

}
```

W.6 Example of xlet Identity Verification

```
/**
 * The following is an example of two xlets verifying eachothers' identities
 * via inter-xlet communication.  When an xlet imports an object from another
 * xlet, the importer knows the organisation ID and application ID of the exporter,
 * but the exporter has no way of verifying the identity of the importer.  If a
 * service were exported without verifying identities, some kinds of spoofing
 * attacks might be possible.  These can be protected against with code modeled
 * on the following Java-like pseudocode.  In this pseudocode, failures (e.g.
 * exceptions) are not addressed.
 **/

public interface VerifiedExporter extends java.rmi.Remote {
    public void acceptConnection(VerifiedImporter im, String orgId, Stirng appId);
}

public interface VerifiedImporter extends java.rmi.Remote {
    public void verifyExporter(VerifiedExporter ex);
}

/**
 * The exporting xlet binds a singleton instance of VerifiedExporterImpl to
 * the name "dvb:/ixc/org_id/app_id/Exporter", where VerifiedExporterImpl contains
 * the following:
 **/

public class VerifiedExporterImpl implements VerifiedExporter {

    public void acceptConnection(VerifiedImporter im, String appId, String orgId) {
        javax.microedition.xlet.ixc.IxcRegistry reg = ... this xlet's registry ...;
        ...  verify orgId and appId represent someone we're willing to talk to ...
        String name = "dvb:/ixc/" + appId + "/" + orgId + "/Importer";
        VerifiedImporter regImp = (VerifiedImporter) reg.lookup(name);
        if (regImp != im) {
            throw new RuntimeException("Spoofing attack foiled");
        }
        ...  Add imp to a collection of objects we're willing to talk to ...
        im.verifyExporter(this);
    }
}

/**
 * The importing xlet binds a singleton instance of VerifiedImporterImpl to
 * the name "dvb:/ixc/org_id/app_id/Importer", where VerifiedImporterImpl contains
 * the following:
 **/

public class VerifiedImporterImpl implements VerifiedImporter {

    private VerifiedExporter peer;

    private VerifiedImporterImpl(VerifiedExporter peer) {
        this.peer = peer;
    }

    public static setupConnection(VerifiedExporter ex) {
        javax.microedition.xlet.ixc.IxcRegistry reg = ... this xlet's registry ...;
        String nm = ... the name of the VerifiedExporter instance ...;
        VerifeedExporter ex = (VerifiedExporter) reg.lookup(name);
        VerifiedImporter im = new VerifiedImporter(ex);
        String orgId = our organisation ID;
        String appId =  our application ID;
        String name = "dvb:/ixc/" + appId + "/" + orgId + "/Importer";
        reg.bind(name, im);
        ex.acceptConnection(im, orgId, appId);
```

```
    }

    public void verifyExporter(VerifiedExporter ex) {
        if (ex != peer) {
            throw new RuntimeException("Spoofing attack foiled");
        }
        ...  Add ex to a collection of objects we're willing to talk to ...
    }
}

/**
 * Note that this technique relies on the == relation being true for an object
 * reference that's sent to a different xlet, then sent back.  This is guaranteed to be
 * true for IXC, but is not a feature of RMI.  For this reason, the above technique
 * will not work with RMI.
 **/
```

Annex X (normative): Test support

GEM [1], annex X is included in the present document.

Annex Y (normative): Inter-application and Inter-Xlet communication API

GEM [1], annex Y is included in the present document with the following modifications:

- The two cautionary notes added to the descriptions of `org.dvb.io.IxcRegistry` and `org.dvb.io.IxcRegistry.lookup()` do not apply to MHP.

Annex Z (informative): Services, Service Contexts and Applications in an MHP Environment

GEM [1], annex Z is included in the present document, with the following notes and modifications:

- This informative clause includes references to some signalling details that, while not required by GEM, are considered normative in MHP.

Annex AA (normative): DVB-HTML 1.0 DTD

AA.1 DVB-HTML 1.0 DTD

AA.1.1 DVB-HTML DTD driver

This clause contains the driver for the DVB-HTML 1.0 document type implementation as an XML DTD. It relies upon XHTML module implementations defined in Modularization [32] and in this annex. This driver is the DTD that shall be referenced by conformant DVB-HTML 1.0 documents.

```
<!-- ....................................................................... -->
<!--                        DVB-HTML 1.0 DTD                               -->
<!-- ....................................................................... -->
<!-- file: dvbhtml-1-0.dtd                                                 -->
<!-- ....................................................................... -->

<!--      This is the DTD driver for DVB-HTML 1.0.

     The following formal public identifier shall be used to identify it:
            "-//DVB//DTD XHTML DVB-HTML 1.0//EN"

    The following URL may be used to reference this file :
        "http://www.dvb.org/mhp/dtd/dvbhtml-1-0.dtd"               -->

<!-- ....................................................................... -->
<!--                       XHTML Driver Parameters                         -->
<!-- ....................................................................... -->

<!ENTITY % XHTML.version "-//DVB//DTD XHTML DVB-HTML 1.0//EN" >
<!-- reserved for use with document profiles                               -->
<!ENTITY % XHTML.profile "" >

<!-- ....................................................................... -->
<!--                              Framework                                -->
<!-- ....................................................................... -->

<!-- Tell the XHTML Framework module to use the DVB-HTML Qualified Names     -->
<!-- module as an extra qname driver                                        -->
<!ENTITY % xhtml-qname-extra.mod
    PUBLIC "-//DVB//ENTITIES DVB-HTML Qualified Names 1.0//EN"
    "dvb-qname-1.mod" >

<!-- Define the Content Model for the framework to use -->
<!ENTITY % xhtml-model.mod
    PUBLIC "-//DVB//ENTITIES DVB-HTML Content Model 1.0//EN"
    "dvbhtml-model-1-0.mod" >

<!-- Include bidirectional text support -->
<!ENTITY % XHTML.bidi "INCLUDE" >

<!-- Comment : Needed for tool verification purpose -->
<!ENTITY % iframe.qname "iframe" >

<!-- XHTML Inline Style Module  ..................................... -->
<!-- needs to be included prior to the XHTML Modular Framework to be taken  -->
<!-- into account                                          -->
<!ENTITY % xhtml-inlstyle.module "INCLUDE">
<![%xhtml-inlstyle.module;[
    !ENTITY % xhtml-inlstyle.mod
    PUBLIC "-//W3C//ENTITIES XHTML Inline Style 1.0//EN"
"http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-inlstyle-1.mod" >
%xhtml-inlstyle.mod;]]>


<!-- Bring in the XHTML Framework module -->
<!ENTITY % xhtml-framework.mod
    PUBLIC "-//W3C//ENTITIES XHTML Modular Framework 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-framework-1.mod"  >
```

```
%xhtml-framework.mod;

<!-- Using tables, so we need to declare this. Is also present on the Content Model -->
<!ENTITY % Table.class "| %table.qname;">

<!-- instantiate the DVB Character Entities module (for the VK_* codes).      -->
<!--
    NOTE 1: The content of this module is not defined in the present document
    since the mappings of those DVB character entities to the character
    set are implementation specific

    NOTE 2: The DVB-HTML Character Entities Module is referenced from the
    DVB-HTML DTDs using a local URL (see <xref AA.1.1 "DVB-HTML DTD Driver").
    Hence the location from which the DTD is fetched will affect the
    values given to the "VK_*" entities defined for use with the HTML
    "accesskey" attribute, described in <xref to 8.5.3.1.8 "Accesskey
    attribute">.  Implementations may therefore choose to store these DTDs
    locally and access them in an implementation-defined manner.
                                                                -->
<!ENTITY % dvb-charent.module "INCLUDE" >
<![%dvb-charent.module;[
<!ENTITY % dvb-charent.mod
    SYSTEM "dvb-charent-1.mod" >
%dvb-charent.mod;]]>

<!-- ..................................................................... -->
<!--                          Modules List                                 -->
<!-- ..................................................................... -->

<!-- This DTD will accept both frameset and body elements as root element childs -->
<!ENTITY % html.content  "( %head.qname;, (%frameset.qname; | %body.qname; ))" >

<!-- XHTML Document Structure Module ..................................... -->
<!ENTITY % xhtml-struct.module "INCLUDE">
<![%xhtml-struct.module;[
<!ENTITY % xhtml-struct.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Document Structure 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-struct-1.mod" >
%xhtml-struct.mod;]]>

<!-- XHTML Text Module ................................................... -->
<!-- Fix up the blockquote content model -->
<!ENTITY % blockquote-fixed.element "INCLUDE">
<![%blockquote-fixed.element;[
<!ENTITY % blockquote.content
     "( #PCDATA | %Block.mix;)*"
>
<!ENTITY % blockquote.qname  "blockquote" >
<!ELEMENT %blockquote.qname;  %blockquote.content; >
<!-- end of blockquote.element -->]]>
<!ENTITY % xhtml-text.module "INCLUDE">
<![%xhtml-text.module;[
<!ENTITY % xhtml-text.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Text 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-text-1.mod" >
%xhtml-text.mod;]]>

<!-- XHTML Hypertext Module ............................... -->
<!ENTITY % xhtml-hypertext.module "INCLUDE">
<![%xhtml-hypertext.module;[
<!ENTITY % xhtml-hypertext.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Hypertext 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-hypertext-1.mod" >
%xhtml-hypertext.mod;]]>

<!-- XHTML Lists Module ................................... -->
<!ENTITY % xhtml-list.module "INCLUDE">
<![%xhtml-list.module;[
<!ENTITY % xhtml-list.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Lists 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-list-1.mod" >
%xhtml-list.mod;]]>

<!-- XHTML Presentation Module  ................................... -->
<!ENTITY % xhtml-pres.module "INCLUDE">
<![%xhtml-pres.module;[
<!ENTITY % xhtml-pres.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Presentation 1.0//EN"
```

```
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-pres-1.mod" >
%xhtml-pres.mod;]]>

<!-- XHTML BDO Element Module  ..................................... -->
<!ENTITY % xhtml-bdo.module "INCLUDE">
<![%xhtml-bdo.module;[
<!ENTITY % xhtml-bdo.mod
    PUBLIC "-//W3C//ELEMENTS XHTML BDO Element 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-bdo-1.mod" >
%xhtml-bdo.mod;]]>

<!-- XHTML Forms Module  ..................................... -->
<!ENTITY % xhtml-form.module "INCLUDE">
<![%xhtml-form.module;[
<!ENTITY % xhtml-form.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Forms 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-form-1.mod" >
%xhtml-form.mod;]]>

<!-- XHTML Basic Tables Module  ..................................... -->
<!ENTITY % xhtml-basic-table.module "INCLUDE">
<![%xhtml-basic-table.module;[
<!ENTITY % xhtml-basic-table.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Basic Tables 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-basic-table-1.mod">
%xhtml-basic-table.mod;]]>

<!-- XHTML Images module ..................................... -->
<!ENTITY % xhtml-image.module "INCLUDE">
<![%xhtml-image.module;[
<!ENTITY % xhtml-image.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Images 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-image-1.mod" >
%xhtml-image.mod;]]>

<!-- XHTML Client Side Image Map module ..................... -->
<!ENTITY % xhtml-csismap.module "INCLUDE">
<![%xhtml-csismap.module;[
<!ENTITY % xhtml-csismap.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Client Side Image Maps 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-csismap-1.mod" >
%xhtml-csismap.mod;]]>

<!-- XHTML Embedded Object module ..................................... -->
<!ENTITY % xhtml-object.module "INCLUDE">
<![%xhtml-object.module;[
<!ENTITY % xhtml-object.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Embedded Object 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-object-1.mod" >
%xhtml-object.mod;]]>

<!-- XHTML Frames module ..................................................-->
<!ENTITY % xhtml-frames.module "INCLUDE">
<![%xhtml-frames.module;[
<!ENTITY % xhtml-frames.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Frames 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-frames-1.mod" >
%xhtml-frames.mod;]]>

<!-- XHTML Target module ..................................... -->
<!ENTITY % xhtml-target.module "INCLUDE">
<![%xhtml-target.module;[
<!ENTITY % xhtml-target.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Target 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-target-1.mod" >
%xhtml-target.mod;]]>

<!-- XHTML IFrame module ..................................... -->
<!ENTITY % xhtml-iframe.module "INCLUDE">
<![%xhtml-iframe.module;[
<!ENTITY % xhtml-iframe.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Inline Frame Element 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-iframe-1.mod" >
%xhtml-iframe.mod;]]>

<!-- DVB-HTML Intrinsic Events Module ..................... -->
<!ENTITY % dvbhtml-events.module "INCLUDE" >
<![%dvbhtml-events.module;[
```

```
<!ENTITY % dvbhtml-events.mod
    PUBLIC "-//DVB//ENTITIES DVB-HTML Intrinsic Events 1.0//EN"
    "http://www.dvb.org/mhp/dtd/dvbhtml-events-1.mod" >
%dvbhtml-events.mod;]]>

<!-- XHTML Document Metainformation Module ........................... -->
<!ENTITY % xhtml-meta.module "INCLUDE">
<![%xhtml-meta.module;[
<!ENTITY % xhtml-meta.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Metainformation 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-meta-1.mod" >
%xhtml-meta.mod;]]>

<!-- XHTML Document Scripting Module  ................................. -->
<!ENTITY % xhtml-script.module "INCLUDE">
<![%xhtml-script.module;[
<!ENTITY % xhtml-script.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Scripting 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-script-1.mod" >
%xhtml-script.mod;]]>

<!-- XHTML Document Stylesheet Module  ................................ -->
<!ENTITY % xhtml-style.module "INCLUDE">
<![%xhtml-style.module;[
<!ENTITY % xhtml-style.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Stylesheets 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-style-1.mod" >
%xhtml-style.mod;]]>

<!-- XHTML Link Element Module  ................................... -->
<!ENTITY % xhtml-link.module "INCLUDE">
<![%xhtml-link.module;[
<!ENTITY % xhtml-link.mod
PUBLIC "-//W3C//ELEMENTS XHTML Link Element 1.0//EN"
"http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-link-1.mod" >
%xhtml-link.mod;]]>

<!-- XHTML Base Element Module  ................................... -->
<!ENTITY % xhtml-base.module "INCLUDE">
<![%xhtml-base.module;[
    !ENTITY % xhtml-base.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Base Element 1.0//EN"
"http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-base-1.mod" >
%xhtml-base.mod;]]>

<!-- ............................................................................ -->
<!-- XHTML Modularization modules ignored in DVB-HTML 1.0 DTD..................... -->
<!-- ............................................................................ -->

<!-- XHTML Java Applet Module  ................................... -->
<!ENTITY % xhtml-applet.module "IGNORE">
<![%xhtml-applet.module;[
    !ENTITY % xhtml-applet.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Java Applets 1.0//EN"
"http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-applet-1.mod" >
%xhtml-applet.mod;]]>

<!-- XHTML Edit Module  ................................... -->
<!ENTITY % xhtml-edit.module "IGNORE">
<![%xhtml-edit.module;[
    !ENTITY % xhtml-edit.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Editing Markup 1.0//EN"
"http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-edit-1.mod" >
%xhtml-edit.mod;]]>

<!-- XHTML Basic Forms Module  ................................... -->
<!ENTITY % xhtml-basic-form.module "IGNORE">
<![%xhtml-basic-form.module;[
    !ENTITY % xhtml-basic-form.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Basic Forms 1.0//EN"
"http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-basic-form-1.mod" >
%xhtml-basic-form.mod;]]>

<!-- XHTML Table Module  ................................... -->
<!ENTITY % xhtml-table.module "IGNORE">
<![%xhtml-table.module;[
    !ENTITY % xhtml-table.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Tables 1.0//EN"
```

```
"http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-table-1.mod" >
%xhtml-table.mod;]]>

<!-- XHTML Server Side Image Map Module  ..................................... -->
<!ENTITY % xhtml-ssismap.module "IGNORE">
<![%xhtml-ssismap.module;[
    !ENTITY % xhtml-ssismap.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Server-side Image Maps 1.0//EN"
"http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-ssismap-1.mod" >
%xhtml-ssismap.mod;]]>

<!-- XHTML Intrinsic Events Module  ..................................... -->
<!ENTITY % xhtml-events.module "IGNORE">
<![%xhtml-events.module;[
    !ENTITY % xhtml-events.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Intrinsic Events 1.0//EN"
"http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-events-1.mod" >
%xhtml-events.mod;]]>

<!-- XHTML Name Identification Module  ..................................... -->
<!ENTITY % xhtml-nameident.module "IGNORE">
<![%xhtml-nameident.module;[
    !ENTITY % xhtml-nameident.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Name Identifier 1.0//EN"
"http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-nameident-1.mod" >
%xhtml-nameident.mod;]]>

<!-- XHTML Legacy Markup Module  ..................................... -->
<!ENTITY % xhtml-legacy.module "IGNORE">
<![%xhtml-legacy.module;[
    !ENTITY % xhtml-legacy.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Legacy Markup 1.0//EN"
"http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-legacy-1.mod" >
%xhtml-legacy.mod;]]>


<!-- end of DVB-HTML 1.0 DTD  ............................................... -->
<!-- ....................................................................... -->
```

AA.1.2 DVB-HTML DVB Intrinsic Events module

This DVB defined module adds some event attributes to the <body> and <frame> element.

```
<!-- ....................................................................... -->
<!--               DVB-HTML Intrinsic Events 1.0 Module                    -->
<!-- ....................................................................... -->
<!-- file: dvbhtml-events-1.mod                                            -->
<!-- ....................................................................... -->

<!--  This is the DTD module for the DVB-HTML defined events.

      The following formal public identifier shall be used to identify it:
         "-//DVB//ENTITIES DVB-HTML Intrinsic Events 1.0//EN"

    The following URL may be used to reference this file :
        "http://www.dvb.org/mhp/dtd/dvbhtml-events-1.mod"

    The namespace and the default prefix of this module are respectively
        xmlns:dvbhtml="http://www.dvb.org/mhp/"
                                                                          -->

<!-- Declare PEs for the DVB Intrinsic events attributes on the body element -->
<!ENTITY % dvbhtml.body.onload.qname    "%dvbhtml.pfx;onload" >
<!ENTITY % dvbhtml.body.onunload.qname  "%dvbhtml.pfx;onunload" >
<!ENTITY % dvbhtml.body.ondvbdomstable.qname "%dvbhtml.pfx;ondvbdomstable" >

<!-- Declare PEs for the DVB-HTML Intrinsic events attributes on the         -->
<!-- frameset element (only used in the DVB-HTML Frameset DTD)                -->
<!ENTITY % dvbhtml.frameset.onload.qname        "%dvbhtml.pfx;onload" >
<!ENTITY % dvbhtml.frameset.onunload.qname  "%dvbhtml.pfx;onunload" >
<!ENTITY % dvbhtml.frameset.ondvbdomstable.qname    "%dvbhtml.pfx;ondvbdomstable" >
```

```
<!-- additional attributes on body element -->
<!ATTLIST %body.qname;
    %dvbhtml.body.onload.qname       %Script.datatype;      #IMPLIED
    %dvbhtml.body.onunload.qname     %Script.datatype;      #IMPLIED
    %dvbhtml.body.ondvbdomstable.qname  %Script.datatype;   #IMPLIED
>

<!-- end of dvbhtml-events-1.mod ..................................... -->
<!-- ................................................................. -->
```

AA.1.3 DVB-HTML Qualified Names module

This DVB-defined module is declared in the DVB-HTML DTD driver and instantiated through the XHTML Modular Framework. It defines parameter entities to support namespace-qualified names. In particular, it defines namespace declarations and name prefixing rules. The actual namespace defined by this module is provided by the following URI: `"http://www.dvb.org/mhp/"` and the default prefix by the string `"dvbhtml"`.

It is defined below:

```
<!-- ................................................................. -->
<!--               DVB-HTML 1.0 Qname Module                   -->
<!-- ................................................................. -->
<!-- file: dvbhtml-qname-1.mod                                 -->
<!-- ................................................................. -->

<!--
     The following formal public identifier shall be used to identify it:
         "-//DVB//ENTITIES DVB-HTML Qualified Names 1.0//EN"

    The following URL may be used to reference this file :
        "http://www.dvb.org/mhp/dtd/dvbhtml-qname-1.mod"             -->


<!-- Bring in the XHTML datatypes, since the URI.datatype parameter entity  -->
<!-- is used for declaring the xmlns attributes.                      -->
<!ENTITY % dvb-datatypes.mod
    PUBLIC "-//W3C//ENTITIES XHTML Datatypes 1.0//EN"
    "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-datatypes-1.mod" >
%dvb-datatypes.mod;

<!-- Declare the default value for prefixing of this module's attributes    -->
<!-- Note that the NS.prefixed will get overridden by the XHTML Framework   -->
<!-- or by a document instance.                                 -->
<!ENTITY % NS.prefixed "IGNORE" >
<!ENTITY % dvbhtml.prefixed "%NS.prefixed;" >

<!-- Declare the actual namespace of this module -->
<!ENTITY % dvbhtml.xmlns "http://www.dvb.org/mhp/" >

<!-- Declare the default prefix for this module -->
<!ENTITY % dvbhtml.prefix "dvbhtml" >

<!-- Declare the prefix and any prefixed namespaces that are required by    -->
<!-- this module. -->
<![%dvbhtml.prefixed;[
<!ENTITY % dvbhtml.pfx "%dvbhtml.prefix;:" >
<!ENTITY % dvbhtml.xmlns.extra.attrib
    "xmlns:%dvbhtml.prefix; %URI.datatype; #FIXED '%dvbhtml.xmlns;'" >
]]>
<!ENTITY % dvbhtml.pfx "" >

<!-- Declare a Parameter Entity (PE) that defines any external namespaces  -->
<!-- that are used by this module                              -->
<!ENTITY % dvbhtml.xmlns.extra.attrib "" >

<!-- Declare a PE that defines the xmlns attributes for use by dvbhtml.    -->
<![%dvbhtml.prefixed;[
<!ENTITY % dvbhtml.xmlns.attrib
    "xmlns:%dvbhtml.prefix; %URI.datatype; #FIXED  %dvbhtml.xmlns;
    %dvbhtml.xmlns.extra.attrib;"
>
]]>
<!ENTITY % dvbhtml.xmlns.attrib
    "xmlns %URI.datatype; #FIXED  %dvbhtml.xmlns;
    %dvbhtml.xmlns.extra.attrib;"
```

```
>
<!-- Make sure that the dvbhtml namespace attributes are included on the   -->
<!-- XHTML attribute set                              -->
<![%NS.prefixed;[
<!ENTITY % XHTML.xmlns.extra.attrib
    "%dvbhtml.xmlns.attrib;" >
]]>
<!ENTITY % XHTML.xmlns.extra.attrib "" >

<!-- end of dvbhtml-qname-1.mod ........................................... -->
<!-- ..................................................................... -->
```

AA.1.4 DVB-HTML Content Model module

The DVB-HTML content model module works together with the DVB-HTML 1.0 DTD driver to customize the XHTML module implementations to the document type's specific requirements. This module is instantiated by the XHTML modular framework module.

It is defined below:

```
<!-- ..................................................................... -->
<!--            DVB-HTML 1.0 Document Model Module                    -->
<!-- ..................................................................... -->
<!-- file: dvbhtml-model-1-0.mod                                 -->
<!-- ..................................................................... -->

<!--  This is the DTD module for the DVB-HTML Document Model.

      The following formal public identifier shall be used to identify it:
             PUBLIC "-//DVB//ENTITIES DVB-HTML Document Model 1.0//EN"

    The following URL may be used to reference this file :
        "http://www.dvb.org/mhp/dtd/dvbhtml-model-1-0.mod"           -->

<!-- Document Model

       This module describes the groupings of elements that make up
       common content models for DVB-HTML elements.                          -->

<!-- ..............  Optional Elements in head  .................. -->

<!ENTITY % HeadOpts.mix   "(        %script.qname;
                                  | %style.qname;
                                  | %meta.qname;
                                  | %link.qname;
                                  | %object.qname;)*" >

<!-- .................  Miscellaneous Elements  .................. -->

<!-- script and noscript are used to contain scripts -->

<!ENTITY % Misc.extra           " | %script.qname;
                                  | %noscript.qname; " >

<!-- These elements are neither block nor inline, and can
              essentially be used anywhere in the document body. -->
<!ENTITY % Misc.class            " %Misc.extra; "  >

<!-- ....................  Inline Elements  ..................... -->

<!ENTITY % InlStruct.class      "   %br.qname;
                                  | %span.qname; " >

<!ENTITY % InlPhras.class       " | %em.qname;
                                  | %strong.qname;
                                  | %dfn.qname;
                                  | %code.qname;
                                  | %samp.qname;
                                  | %kbd.qname;
                                  | %var.qname;
                                  | %cite.qname;
                                  | %abbr.qname;
                                  | %acronym.qname;
                                  | %q.qname; " >
```

```
<!ENTITY % InlPres.class        " | %tt.qname;
                                  | %i.qname;
                                  | %b.qname;
                                  | %big.qname;
                                  | %small.qname;
                                  | %sub.qname;
                                  | %sup.qname; " >

<!ENTITY % I18n.class           " | %bdo.qname; " >

<!ENTITY % Anchor.class         " | %a.qname; " >

<!ENTITY % InlSpecial.class     " | %img.qname;
                                  | %map.qname;
                                  | %object.qname; " >

<!ENTITY % Control.class        " | %input.qname;
                                  | %select.qname;
                                  | %textarea.qname;
                                  | %label.qname;
                                  | %button.qname; " >


<!ENTITY % Inline.extra         " | %iframe.qname; " >


<!-- %Inline.class; includes all inline elements, used as a component in mixes -->
<!ENTITY % Inline.class           " %InlStruct.class;
                                    %InlPhras.class;
                                    %InlPres.class;
                                    %I18n.class;
                        %Control.class;
                                    %Anchor.class;
                                    %InlSpecial.class;
                                    %Inline.extra;" >

<!-- %InlNoAnchor.class; includes all non-anchor inline elements,
used as a component in mixes -->

<!ENTITY % InlNoAnchor.class      " %InlStruct.class;
                                    %InlPhras.class;
                                    %InlPres.class;
                                    %I18n.class;
                        %Control.class;
                                    %InlSpecial.class;
                                    %Inline.extra;" >

<!-- %Inline-noa.mix; includes all non-anchor inlines -->
<!ENTITY % InlNoAnchor.mix        " %InlNoAnchor.class;
                                    %Misc.class;">

<!-- %Inline.mix; includes all inline elements, including %Misc.class; -->
<!ENTITY % Inline.mix             " %Inline.class;
                                    %Misc.class;" >


<!-- .....................  Block Elements  ..................... -->

<!ENTITY % Heading.class          " %h1.qname;
                                  | %h2.qname;
                                  | %h3.qname;
                                  | %h4.qname;
                                  | %h5.qname;
                                  | %h6.qname; " >

<!ENTITY % List.class             " %ul.qname;
                                  | %ol.qname;
                                  | %dl.qname; " >

<!ENTITY % BlkStruct.class        " %p.qname;
                                  | %div.qname; " >

<!ENTITY % BlkPhras.class       " | %pre.qname;
                                  | %blockquote.qname;
                                  | %address.qname; " >

<!ENTITY % BlkPres.class         "| %hr.qname; " >
```

```
<!ENTITY % BlkNoTable.extra      "| %form.qname;
                                  | %fieldset.qname; " >

<!ENTITY % Table.class          "| %table.qname; " >


<!ENTITY % Block.extra             "%Table.class;
                                   | %form.qname;
                                   | %fieldset.qname; " >

<!ENTITY % BlkNoTable.class        "%BlkStruct.class;
                                    %BlkPhras.class;
                                    %BlkPres.class;
                                    %BlkNoTable.extra;" >


<!-- %Block.class; includes all block elements, used as a component in mixes -->
<!ENTITY % Block.class             "%BlkStruct.class;
                                    %BlkPhras.class;
                                    %BlkPres.class;
                                    %Block.extra;" >

<!-- %Block.mix; includes all block elements plus %Misc.class; -->
<!ENTITY % Block.mix               "%Heading.class;
                                   | %List.class;
                                   | %Block.class;
                                    %Misc.class;" >

<!-- ................  All Content Elements  .................. -->

<!ENTITY % FlowNoTable.mix         "%Heading.class;
                                   | %List.class;
                                   | %BlkNoTable.class;
                                   | %Inline.class;
                                    %Misc.class;" >


<!-- %Flow.mix; includes all text content, block and inline -->
<!ENTITY % Flow.mix                "%Heading.class;
                                   | %List.class;
                                   | %Block.class;
                                   | %Inline.class;
                                    %Misc.class;" >

<!-- ....................................................................... -->
<!-- end of dvbhtml-model-1-0.mod                                           -->
<!-- ....................................................................... -->
```

Annex AB (normative): DVB HTML StyleSheet

In the absence of any other styling information for a page, the support application shall render using the following default stylesheet:

```
/*
  * Since MHP implementations may add element types, this rule prevents
  * any such elements from being displayed by default.  The following
  * rule restores the default "inline" value to HTML element types which
  * should be displayed inline.
  */
*               { display: none }
style, meta, link,
script, noscript,
br, span,
em, strong, dfn,
code, samp, kbd,
var, cite, abbr,
acronym, q,
tt, i, d, big,
small, sub, sup,
bdo, a,
img, map, object,
input, select,
textarea, label,
button          { display: inline }

address,
blockquote,
body, dd, div,
dl, dt, fieldset,
form, frame,
frameset, h1, h2,
h3, h4, h5, h6,
hr, noframes, ol,
p, pre, ul      { display: block }
object          { display: inline }
li              { display: list-item }
head            { display: none }
table           { display: table }
tr              { display: table-row }

/*
* NOTE: Removed for DVB-HTML, as these elements and these values of display
* property are not supported.
*
* thead          { display: table-header-group }
* tbody          { display: table-row-group }
* tfoot          { display: table-footer-group }
* col            { display: table-column }
* colgroup       { display: table-column-group }
*/
td, th          { display: table-cell }
caption         { display: table-caption }
th              { font-weight: bolder; text-align: center }
caption         { text-align: center }

/*
* NOTE: The padding here and the initial block below (in the @viewport
* rule) is defined such that the initial block matches the "Safe Action
* Area" and the content area of the BODY matches the "Safe Title Area".
* The Safe Action Area is the centre 90% of the screen; the Safe Title
* Area is the centre 80%. Percentages are not used to specify the body
* padding because both axes are interpreted with respect to the *width*
* of the containing block. Therefore the padding below is in pixels.
*
* NOTE: The sample HTML 4.0 stylesheet in the CSS2 specification, derived
* from a survey of existing User Agents, has "line-height: 1.33". However,
* the errata (as of 19980512) reduce this to 1.12em.
*/
body            { font-family: Tiresias; font-size: 26pt;
                  padding: 29px 36px; line-height: 1.12em;
```

```
                  opacity: 0;
                  background-color: rgb(0, 0, 0);
                  color: rgb(255, 255, 255); }

/*
* NOTE: The margins here are chosen to be 26pt/18px (according to the
* information on Tiresias in TAM232r16). They are
* expressed in "em" units so overriding stylesheets can change just the
* font-size and have the margins scaled.
*/
h6, h4, h2        { text-transform: uppercase }
h6                { font-size: 24pt; margin: 1.08em 0 }
h5, h4            { font-size: 26pt; margin: 1em 0 }
h3, h2            { font-size: 31pt; margin: 0.84em 0 }
h1                { font-size: 36pt; margin: 0.72em 0 }

/*
* NOTE: The vertical margin here is left at 1.33em despite the CSS 2 errata
* mentioned above.
*/
blockquote, dl, fieldset, form, ol, p, ul { margin: 1.33em 0 }
h1, h2, h3, h4, h5, h6,
strong            { font-weight: bolder }
blockquote        { margin-left: 1.5em; margin-right: 1.5em }
address, cite,
em, i, var        { font-style: italic }
pre, code, kbd,
samp              { font-family: monospace }
pre               { white-space: pre }
big               { font-size: larger }
small, sub, sup   { font-size: smaller }
sub               { vertical-align: sub }
sup               { vertical-align: super }
hr                { border: 1px solid }
ol, ul, dd        { margin-left: 1.5em }
ol                { list-style-type: decimal }
ol ul, ul ol,
ul ul, ol ol      { margin-top: 0; margin-bottom: 0 }

br:before         { content: "\A" }

abbr, acronym     { letter-spacing: 0.1em }

:link             { color: rgb(0, 127, 255) } /* opaque blue */
:focus            { color: rgb(0, 127, 255); /* opaque blue */
                    outline: 2px solid rgb(255, 0, 127) } /* opaque blue */
:active           { color: rgb(255, 31, 0); /* opaque red */
                    outline: 2px solid rgb(255, 31, 0) } /* opaque red */
:visited          { color: rgb(191, 0, 127) } /* opaque violet */

/* Begin bidirectionality settings (do not change) */
bdo[dir="ltr"]    { direction: ltr; unicode-bidi: bidi-override }
bdo[dir="rtl"]    { direction: rtl; unicode-bidi: bidi-override }

*[dir="ltr"]      { direction: ltr; unicode-bidi: embed }
*[dir="rtl"]      { direction: rtl; unicode-bidi: embed }

/* Elements that are block-level in HTML4 */
/*
* NOTE: Removed for DVB-HTML, as these elements and these values of display
* property are not supported: thead, tbody, tfoot, col, colgroup
*/
address, blockquote, body, dd, div, dl, dt, fieldset,
form, frame, frameset, h1, h2, h3, h4, h5, h6, iframe,
noscript, noframes, object, ol, p, ul,
hr, pre, li, table, tr, td, th, caption { unicode-bidi: embed }
/* End bidi settings */
```

```
@viewport {
  /* scene: defaults to rect(0%, 0%, 100%, 100%), full screen */
  /* horizontal-resolution: defaults to 720px; */
  /* vertical-resolution: 576px; */
  /* By default, stay within Safe Action Area (centre 90%) */
  /* initial: (36px, 29px, 648px, 518px); */
  initial: rect(10%, 10%, 90%, 90%);
  type: none;
  background: rgb(0, 0, 0);
  area: screen;
}
```

Annex AC (normative): ECMAScript Binding

This appendix contains the complete ECMA-262 [33] binding for the DVB HTML Document Object Model ECMAScript Language Binding.

AC.1 ECMAScript language binding

Since ECMAScript is a prototype-based language, the DOM implementation using this language can add mechanisms which can not be defined using the IDL language. This clause defines two sets of functionalities that an ECMAScript DOM DVB implementation shall support.

For other considerations concerning the ECMAScript language binding, please refer to the appendix D of HTML 4 [41].

AC.1.1 Shortcuts to access objects

The IDL interfaces defined in clause 8.11.4.6, "DVB-HTML element related interfaces" provide access to some objects using DVB HTML collections. For example, the third form element of a document can be accessed by using the following instruction, where `document` is the name of the object representing the DVB HTML document:

```
document.forms[2]
```

In this perspective, each object shall add a new property for each object that it encapsulates. The name of the property shall be the identifier of the corresponding encapsulated object (i.e. the value of the `id` attribute inherited by the `DVBHTMLElement` interface). The document will fail to parse if the id of an element would rename a property of the host object. Setting a property on the ECMAScript GlobalObject will override the shortcut of the same name.

This property only exists if the element has an id attribute. This shall be applied for the following interfaces:

- `DVBHTMLAnchorElement;`
- `DVBHTMLMapElement;`
- `DVBHTMLAreaElement;`
- `DVBHTMLButtonElement;`
- `DVBHTMLFrameElement;`
- `DVBHTMLFramesetElement;`
- `DVBHTMLImageElement;`
- `DVBHTMLInputElement;`
- `DVBHTMLOptionElement;`
- `DVBHTMLSelectElement;`
- `DVBHTMLTextareaElement.`

For example, if a document contains a form whose identifier is `myForm`, the form can be accessed using the following equivalent instructions:

```
document.myForm
document["myForm"]
myForm // since document is the global object
```

Subsequently, if the form contains a button identified by the `myButton` string, the button can be accessed using:

```
document.myForm.myButton
```

AC.1.2 Grouping the objects of a form

In DVB HTML, the following elements of a form may belong to different groups: button, input, select, textarea. The `name` attribute of those elements identifies this group. The `DVBHTMLFormElement` interface does not represent this relation. Instead, all the elements included in the form are accessed by the `elements` collection. To determine the elements belonging to the same group, an application can list all the elements, looking for the ones with the `name` attribute equals to a particular group name.

A DOM DVB HTML implementation shall provide an easier access by integrating a property in a form element for each group included in this form. The property is a table containing all the elements of the group. The name of the property shall be the `name` attribute of the elements which corresponds to the name of the group. For example, if the DVB HTML document contains a form identified by `myForm`, with two elements having their name attribute set to `myGroup`, these buttons will be accessed using the following instructions:

```
document.myForm.myGroup[0]
document.myForm.myGroup[1]
```

The rule defined in the previous clause also applies. Thus, if the two elements are respectively identified by `myInput1`, and `myInput2`, they can also be accessed by the following instructions:

```
document.myForm.myInput1
document.myForm.myInput2
```

Methods that attempt to set read-only attributes shall cause the DOMException with the error code NO_MODIFICATION_ALLOWED_ERR to be raised.

AC.2 The DVB-HTML host objects

AC.2.1 Object DVBHTMLCollection

The DVBHTMLCollection object has the following properties:

length: This read-only property is of type Number.

The DVBHTMLCollection object has the following methods:

item(index): This method returns a Node object.

The **index** parameter is of type Number.

NOTE 1: This object can also be dereferenced using square bracket notation (e.g. obj[1]).

Dereferencing with an integer index is equivalent to invoking the item method with that index.

namedItem(name): This method returns a Node object.

The **name** parameter is of type String.

NOTE 2: This object can also be dereferenced using square bracket notation (e.g. obj["foo"]).Dereferencing using a string index is equivalent to invoking the namedItem method with that name.

NOTE 3: In DVB HTML names are not used for identification purposes, so only Nodes with id attributes will appear in a DVBHTMLCollection object.

See:

- clause 8.11.4.5.1, "DVB-HTMLCollection Interface".

AC.2.2 Object DVBHTMLDocument

DVBHTMLDocument has the all the properties and methods of the Document object as well as the properties and methods defined below.

The DVBHTMLDocument object has the following properties:

title: This property is of type String.

referrer: This read-only property is of type String.

domain: This read-only property is of type String.

URL: This read-only property is of type String.

body: This property is a DVBHTMLElement object.

images: This read-only property is a DVBHTMLCollection object.

links: This read-only property is a DVBHTMLCollection object.

forms: This read-only property is a DVBHTMLCollection object.

anchors: This read-only property is a DVBHTMLCollection object.

cookie: This property is of type String.

The DVBHTMLDocument object has the following methods:

open(): This method has no return value. The ECMAScript context which opened the new document cannot be destroyed until it calls document.close().

close(): This method has no return value. Deletes the ECMAScript context that opened the document.

write(text): This method has no return value.

The **text** parameter is of type String.

writeln(text): This method has no return value.

The **text** parameter is of type String.

See:

- clause 8.11.4.5.2, "DVBHTMLDocument Interface".

AC.2.3 Object DVBHTMLElement

DVBHTMLElement has the all the properties and methods of the Element object as well as the properties and methods defined below.

The DVBHTMLElement object has the following properties:

id: This property is of type String.

title: This property is of type String.

lang: This property is of type String; This property can raise a DOMException on setting.

dir: This property is of type String; This property can raise a DOMException on setting.

className: This property is of type String.

AC.2.4 Object DVBHTMLFormElement

DVBHTMLFormElement has the all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLFormElement object has the following properties:

elements: This read-only property is a DVBHTMLCollection object.

length: This read-only property is of type Number.

acceptCharset: This property is of type String.

action: This property is of type String.

enctype: This property is of type String.

method: This property is of type String; This property can raise a DOMException on setting.

target: This property is of type String.

The DVBHTMLFormElement object has the following methods:

submit(): This method has no return value.

reset(): This method has no return value.

See:

- clause 8.11.4.6.6, "DVBHTMLFormElement Interface".

AC.2.5 Object DVBHTMLSelectElement

DVBHTMLSelectElement has the all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLSelectElement object has the following properties:

type: This read-only property is of type String.

selectedIndex: This property is a long object.

value: This property is of type String.

length: This read-only property is of type Number

form: This read-only property is a DVBHTMLFormElement object.

options: This read-only property is a DVBHTMLCollection object.

disabled: This property is of type Boolean.

multiple: This read-only property is of type Boolean.

name: This property is of type String.

size: This property is of type Number.

The DVBHTMLSelectElement object has the following methods:

add(element, before): This method has no return value.

The **element** parameter is a DVBHTMLElement object.

The **before** parameter is a DVBHTMLElement object.

This method can raise a **DOMException** object.

remove(index): This method has no return value.

The **index** parameter is of type Number.

blur(): This method has no return value.

focus(): This method has no return value.

See:

- clause 8.11.4.6.14, "DVBHTMLSelectElement Interface".

AC.2.6 Object DVBHTMLOptionElement

DVBHTMLOptionElement has the all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLOptionElement object has the following properties:

form: This read-only property is a DVBHTMLFormElement object.

defaultSelected: This property is of type Boolean.

text: This read-only property is of type String.

index: This read-only property is of type Number.

disabled: This property is of type Boolean.

label: This property is of type String.

selected: This property is of type Boolean.

value: This property is of type String.

See:

- clause 8.11.4.6.13, "DVBHTMLOptionElement Interface".

AC.2.7 Object DVBHTMLInputElement

DVBHTMLInputElement has the all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLInputElement object has the following properties:

defaultValue: This property is of type String.

defaultChecked: This property is of type Boolean.

form: This read-only property is a DVBHTMLFormElement object.

accept: This property is of type String.

accessKey: This property is of type String.

alt: This property is of type String.

checked: This property is of type Boolean.

disabled: This property is of type Boolean.

maxLength: This property is of type Number.

name: This property is of type String.

readOnly: This property is of type Boolean.

size: This property is of type String.

src: This property is of type String.

type: This read-only property is of type String.

useMap: This read-only property is of type String.

value: This property is of type String.

The DVBHTMLInputElement object has the following methods:

blur(): This method has no return value.

focus(): This method has no return value.

select(): This method has no return value.

click(): This method has no return value.

See:

- clause 8.11.4.6.12, "DVBHTMLInputElement Interface".

AC.2.8 Object DVBHTMLTextAreaElement

DVBHTMLTextAreaElement has the all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLTextAreaElement object has the following properties:

defaultValue: This property is of type String.

form: This read-only property is a DVBHTMLFormElement object.

accessKey: This property is of type String.

cols: This read-only property is a long object.

disabled: This property is of type Boolean.

name: This property is of type String.

readOnly: This property is of type Boolean.

rows: This property is a long object.

type: This read-only property is of type String.

value: This property is of type String.

The DVBHTMLTextAreaElement object has the following methods:

blur(): This method has no return value.

focus(): This method has no return value.

select(): This method has no return value.

See:

- clause 8.11.4.6.15, "DVBHTMLTextAreaElement Interface".

AC.2.9 Object DVBHTMLButtonElement

DVBHTMLButtonElement has the all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLButtonElement object has the following properties:

form: This read-only property is a DVBHTMLFormElement object.

accessKey: This property is of type String.

disabled: This property is of type Boolean.

name: This property is of type String.

type: This read-only property is of type String.

value: This property is of type String.

See:

- clause 8.11.4.6.5, "DVBHTMLButtonElement Interface".

AC.2.10 Object DVBHTMLAnchorElement

DVBHTMLAnchorElement has the all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLAnchorElement object has the following properties:

accessKey: This property is of type String.

charset: This property is of type String.

href: This property is of type String.

hreflang: This property is of type String.

target: This property is of type String.

type: This property is of type String.

The DVBHTMLAnchorElement object has the following methods:

blur(): This method has no return value.

focus(): This method has no return value.

See:

- clause 8.11.4.6.2, "DVBHTMLAnchorElement Interface".

AC.2.11 Object DVBHTMLImageElement

DVBHTMLImageElement has the all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLImageElement object has the following properties:

lowSrc: This property is of type String.

alt: This property is of type String.

height: This property is of type String.

longDesc: This property is of type String.

src: This property is of type String.

useMap: This read-only property is of type String.

width: This property is of type String.

See:

- clause 8.11.4.6.10, "DVBHTMLImageElement Interface".

AC.2.12 Object DVBHTMLObjectElement

DVBHTMLObjectElement has the all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLObjectElement object has the following properties:

form: This read-only property is a DVBHTMLFormElement object.

code: This property is of type String.

archive: This property is of type String.

codeBase: This property is of type String.

codeType: This property is of type String.

data: This property is of type String.

declare: This property is of type Boolean.

height: This property is of type String.

standby: This property is of type String.

tabIndex: This property is a long object.

type: This property is of type String.

useMap: This read-only property is of type String.

width: This property is of type String.

contentDocument: This read-only property is a Document object.

See:

- clause 8.11.4.6.11, "DVBHTMLObjectElement Interface".

AC.2.13 Object DVBHTMLMapElement

DVBHTMLMapElement has the all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLMapElement object has the following properties:

areas: This read-only property is a DVBHTMLCollection object.

See:

- clause 8.11.4.6.3, "DVBHTMLMapElement Interface".

AC.2.14 Object DVBHTMLAreaElement

DVBHTMLAreaElement has the all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLAreaElement object has the following properties:

accessKey: This property is of type String.

alt: This property is of type String.

href: This property is of type String.

noHref: This property is of type Boolean.

target: This property is of type String.

See:

- clause 8.11.4.6.4, "DVBHTMLAreaElement Interface".

AC.2.15 Object DVBHTMLFrameSetElement

DVBHTMLFrameSetElement has the all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLFrameSetElement object has the following properties:

cols: This read-only property is of type String.

rows: This read-only property is of type String.

See:

- clause 8.11.4.6.8, "DVBHTMLFrameSetElement Interface".

AC.2.16 Object DVBHTMLFrameElement

DVBHTMLFrameElement has the all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLFrameElement object has the following properties:

frameBorder: This read-only property is of type String.

longDesc: This property is of type String.

marginHeight: This read-only property is of type String.

marginWidth: This read-only property is of type String.

scrolling: This read-only property is of type String.

src: This property is of type String.

contentDocument: This read-only property is a Document object.

See:

- clause 8.11.4.6.7, "DVBHTMLFrameElement Interface".

AC.2.17 Object DVBHTMLIFrameElement

DVBHTMLIFrameElement has the all the properties and methods of the DVBHTMLElement object as well as the properties and methods defined below.

The DVBHTMLIFrameElement object has the following properties:

frameBorder: This read-only property is of type String.

longDesc: This property is of type String.

marginHeight: This read-only property is of type String.

marginWidth: This read-only property is of type String.

scrolling: This read-only property is of type String.

src: This property is of type String.

contentDocument: This read-only property is a Document object.

See:

- clause 8.11.4.6.9, "DVBHTMLIFrameElement Interface".

AC.3 DVB-HTML event host objects

AC.3.1 Object DVBLifecycleEvent

DVBLifecycleEvent has the all the properties and methods of the DOMEvent object as well as the properties and methods defined below.

The DVBLifecycleEvent object has the following properties:

detail: This readonly property is of type Number.

The DVBHTMLAnchorElement object has the following methods:

initDVBLifecycleEvent(typeArg, canBubbleArg, cancelableArg, detailArg): This method has no return value.

The **typeArg** parameter is of type String.

The **canBubbleArg** parameter is of type Boolean.

The **cancelableArg** parameter is of type Boolean.

The **detailArg** parameter is of type Number.

See:

- clause 8.11.2.2.1, "Interface DVBLifecycleEvent".

AC.4 DVB-HTML environment host objects

AC.4.1 Navigator Object

appCodeName: This read-only property is of type String.

appName: This read-only property is of type String.

appVersion: This read-only property is of type String.

userAgent: This read-only property is of type String.

See:

- clause 8.11.7.2.1, "Navigator Object".

AC.4.2 Window Object

Window object has the all the properties and methods defined below.

The Window object has the following properties:

document: This read-only property is of type DVBHTMLDocument.

DOMImplementation: This read-only property is of type DVBDOMImplementation.

frames: This read-only property is of type DVBHTMLCollection.

length: This read-only property is of type Number.

location: This property is of type Location.

name: This read-only property is of type String.

navigator: This read-only property is of type Navigator.

parent: This read-only property is of type Window.

window: This read-only property is of type Window.

self: This read-only property is of type Window.

status: This property is of type String.

defaultStatus: This property is of type Strin.

top: This read-only property is of type Window.

The Window object has the following methods:

clearTimeout(timerId): This method has no return value.

The parameter **timerId** is of type Number.

open(url, name, features): This method returns a Window object.

The parameter **url** is of type String.

The parameter **name** is of type String.

The parameter **features** is of type String.

setTimeout(statement, delay): This method returns a Number.

The **statement** parameter is of type String.

The **delay** parameter is of type Number.

See:

- clause 8.11.7.2.2, "Window object".

AC.4.3 Location object

Location has the all the properties and methods defined below.

The Location object has the following properties:

hash: This property is of type String.

host: This property is of type String.

hostname: This property is of type String.

href: This property is of type String.

pathname: This property is of type String.

port: This property is of type String.

protocol: This property is of type String.

search: This property is of type String.

See:

- clause 8.11.7.2.3, "Location object".

AC.5 DVB-HTML CSS host objects

AC.5.1 DVBInlineStyle

The DVBElement object for elements supporting the "style" attribute has the additional properties:

style: This read-only property is of type CSS2Properties.

See:

- clause 8.11.8.1.1, "DVBCSSInlineStyle".

AC.5.2 DVBCSSStyle

The DVBElement object for the root element in a top level document has the additional properties:

viewport: This read-only property is of type DVBCSSViewportRule.

See:

- clause 8.11.8.1.2, "DVBCSSStyle".

AC.5.3 DVBCSSViewportRule

DVBCSSviewportRule has the all the properties and methods of the CSSRule object as well as the properties and methods defined below.

style: This read-only property is of type DVBCSSViewportProperties.

See:

- clause 8.11.8.1.3, "DVBCSSViewportRule".

AC.5.4 DVBCSSViewportProperties

DVBCSSViewportProperties has the following properties:

pseudoClass: This property is of type String.

area: This property is of type String.

backgroundVideoTransform: This property is of type Array of String.

backgroundVideo: This property is of type Array of String.

backgroundVideoPreserveAspect: This property is of type Array of String.

backgroundVideoClip: This property is of type Array of String.

backgroundVideoRectangle: This property is of type Array of String.

background: This property is of type String.

backgroundImageRectangle: This property is of type String.

initial: This property is of type String.

verticalResolution: This property is of type Number.

horizontalResolution: This property is of type Number.

scene: This property is of type String".

See:

- clause 8.11.8.1.4, "DVBCSSViewportProperties".

Annex AD (normative): Support for DVB-HTML

AD.1 Java bindings to DVB extensions

AD.1.1 The org.dvb.dom.dvbhtml package

All of these are in the package `org.dvb.dom.dvbhtml`.

AD.1.1.1 DVBHTMLButtonElement

```
public interface DVBHTMLButtonElement
    extends DVBHTMLElement {
    public String getAccessKey();
    public void setAccessKey(String key);
    public boolean getDisabled();
    public void setDisabled(boolean disabled)
    public DVBHTMLFormElement getForm();
    public String getName();
    public void setName(String name);
    public String getType();
    public void setValue(String value);
    public String getValue();
    public void blur();
    public void focus();
};
```

AD.1.1.2 DVBHTMLCollection

See:

- clause 8.11.4.5.1, "DVB-HTMLCollection Interface";
- clause AC.2.1, "Object DVBHTMLCollection".

```
public interface DVBHTMLCollection {
//Methods
    public Node getIdentifiedItem(String name)
    public Node getItem(Number index)
    public Number getLength()
}
```

AD.1.1.3 DVBHTMLDocument

See:

- clause 8.11.4.5.2, "DVBHTMLDocument Interface";
- clause AC.2.2, "Object DVBHTMLDocument".

```
public interface DVBHTMLDocument
    extends Document {
//Methods
    public DVBHTMLCollection getAnchors()
    public DVBHTMLElement getBody()
    public String getCookie()
    public String getDomain()
    public DVBHTMLCollection getForms()
    public DVBHTMLCollection getImages()
    public DVBHTMLCollection getLinks()
    public String getReference()
    public String getTitle()
    public void setTitle(String title)
```

```
    public String getURL()
    public void open()
        throws IllegalStateException
    public void close()
        throws IllegalStateException
    public void write(String text)
        throws IllegalStateException
    public void writeIn(String text)
        throws IllegalStateException
}
```

AD.1.1.4 DVBHTMLElement

See:

- clause 8.11.4.6.1, "DVBHTMLElement Interface";
- clause AC.2.3, "Object DVBHTMLElement".

```
public interface DVBHTMLElement
    extends Element {
    public String getClassName()
    public void setClassName()
    public String getDir()
    public void setDir()
    public String getId()
    public void setId(String id)
    public String getLang()
    public void setLang(String lang)
    public String getTitle()
    public void setTitle(String)
}
```

AD.1.1.5 DVBHTMLFormElement

See:

- clause 8.11.4.6.6, "DVBHTMLFormElement Interface";
- clause AC.2.4, "Object DVBHTMLFormElement".

```
public interface DVBHTMLFormElement
    extends DVBHTMLElement {
    public String getAction()
    public void setAction(String action)
    public String getCharSet()
    public void setCharSet(String charset)
    public DVBHTMLCollection getElements()
    public String getEncType()
    public void setEncType(String action)
    public long getLength()
    public String getMethod()
    public void setMethod(String action)
    public String getTarget()
    public void setTarget(String action)
    public void submit()
    public void reset()
}
```

AD.1.1.6 DVBHTMLSelectElement

See:

- clause 8.11.4.6.14, "DVBHTMLSelectElement Interface";
- clause AC.2.5, "Object DVBHTMLSelectElement".

```
public interface DVBHTMLSelectElement
    extends DVBHTMLElement {
    public String getType()
    public long getSelectedIndex()
```

```
    public void setSelectedIndex(long index)
    public String getValue()
    public void setvalue(String value)
    public long getLength()
    public DVBHTMLFormElement getForm()
    public DVBHTMLCollection getOptions()
    public boolean isDisabled()
    public void setDisabled(Boolean disabled)
    public boolean isMultiple()
    public String getName()
    public void setName(String name)
    public long getSize()
    public void setSize(long index)
    public long getTabIndex()
    public void setTabIndex(long index)
    public void add(DVBHTMLElement element, DVBHTMLElement before)
        throws DOMException
    public void remove(long index)
    public void blur()
    public void focus()
}
```

AD.1.1.7 DVBHTMLOptionElement

See:

- clause 8.11.4.6.13, "DVBHTMLOptionElement Interface";
- clause AC.2.6, "Object DVBHTMLOptionElement".

```
public interface DVBHTMLOptionElement
    extends DVBHTMLElement {
    public boolean isDefaultSelected()
    public void setDefaultSelected(boolean defaultselected)
    public boolean isSelected()
    public void setSelected(boolean selected)
    public boolean isDisabled()
    public void setDisabled(boolean disabled)
    public DVBHTMLFormElement getForm()
    public long getIndex()
    public String getLabel()
    public void setLabel(String Lable)
    public String getText()
    public void setText(String text)
    public String getValue()
    public void setValue(String value)
}
```

AD.1.1.8 DVBHTMLInputElement

See:

- clause 8.11.4.6.12, "DVBHTMLInputElement Interface";
- clause AC.2.7, "Object DVBHTMLInputElement".

```
public interface DVBHTMLInputElement
    extends DVBHTMLElement {
    public boolean isChecked()
    public void setChecked(boolean checked)
    public boolean isDefaultChecked()
    public void setDefaultChecked(boolean defaultchecked)
    public boolean isDisabled()
    public void setDisabled(boolean disabled)
    public boolean isReadOnly()
    public void setReadOnly(boolean readonly)
    public String getAccept()
    public void setAccept(String accept)
    public String getAccessKey()
    public void setAccessKey(String accesskey)
    public String getAlt()
    public void setAlt(String alt)
    public String getDefaultValue()
    public void setDefaultValue(String value)
```

```
    public DVBHTMLFormElement getForm()
    public long getMaxLength()
    public void setMaxLength(long length)
    public String getName()
    public void setName(String name)
    public String getSize()
    public void setSize(String size)
    public String getSrc()
    public void setSrc(String src)
    public String getType()
    public void setType(String type)
    public String getUseMap()
    public String getValue()
    public void setValue(String value)
    public void blur()
    public void click()
    public void focus()
    public void select()
}
```

AD.1.1.9 DVBHTMLTextAreaElement

See:

- clause 8.11.4.6.15, "DVBHTMLTextAreaElement Interface";
- clause AC.2.8, "Object DVBHTMLTextAreaElement".

```
public interface DVBHTMLTextAreaElement
    extends DVBHTMLElement {
    public boolean isDisabled()
    public void setDisabled(boolean disabled)
    public String getAccessKey()
    public void setAccessKey(String accesskey)
    public String getDefaultValue()
    public void setDefaultValue(String value)
    public DVBHTMLFormElement getForm()
    public String getName()
    public void setName(String name)
    public String getType()
    public void setType(String type)
    public String getValue()
    public void setValue(String value)
}
```

AD.1.1.10 DVBHTMLAnchorElement

See:

- clause 8.11.4.6.2, "DVBHTMLAnchorElement Interface";
- clause AC.2.10, "Object DVBHTMLAnchorElement".

```
public interface DVBHTMLAnchorElement
    extends DVBHTMLElement {
    public String getAccessKey()
    public void setAccessKey(String accesskey)
    public String getCharSet()
    public void setCharSet(String charset)
    public String getHRef()
    public void setHRef(String href)
    public String getHRefLang()
    public void setHRefLang(String hreflang)
    public String getTarget()
    public void setTarget(String target)
    public String getType()
    public void setType(String type)
    public void blur()
    public void focus()
}
```

AD.1.1.11 DVBHTMLImageElement

See:

- clause 8.11.4.6.10, "DVBHTMLImageElement Interface";
- clause AC.2.11, "Object DVBHTMLImageElement".

```
public interface DVBHTMLImageElement
    extends DVBHTMLElement {
    public String getLOwSrc()
    public void setLowSrc(String lowsrc)
    public String getAlt()
    public void setAlt(String alt)
    public String getHeight()
    public void setHeight(String height)
    public String getLongDesc()
    public void setLongDesc(String longdesc)
    public String getArc()
    public void setArc(String arc)
    public String getUsemap()
    public void setUseMap(String usemap)
    public String getWidth()
    public void setWidth(String width)
}
```

AD.1.1.12 DVBHTMLObjectElement

See:

- clause 8.11.4.6.11, "DVBHTMLObjectElement Interface";
- clause AC.2.12, "Object DVBHTMLObjectElement".

```
public interface DVBHTMLObjectElement
    extends DVBHTMLElement {
    public boolean isDeclare()
    public void setDeclare(boolean declare)
    public DVBHTMLFormElement getForm()
    public String getCode()
    public void setCode(String code)
    public String getArchive()
    public void setArchive(String archive)
    public String getCodeBase()
    public void setCodeBase(String codebase)
    public String getCodeType()
    public void setCodeType(String codetype)
    public String getData()
    public void setData(String data)
    public String getHeight()
    public void setHeight(String height)
    public String getStandby()
    public void setStandby(String standby)
    public long getTabIndex()
    public void setTabIndex(String index)
    public String getType()
    public void setType(String type)
    public String getUseMap()
    public void setUseMap(String usemap)
    public String getWidth()
    public void setWidth(String width)
    public Document getContentDocument()
}
```

AD.1.1.13 DVBHTMLMapElement

See:

- clause 8.11.4.6.3, "DVBHTMLMapElement Interface";
- clause AC.2.13, "Object DVBHTMLMapElement".

```
public interface DVBHTMLMapElement
    extends DVBHTMLElement {
    public DVBHTMLCollection getAreas()
}
```

AD.1.1.14 DVBHTMLAreaElement

See:

- clause 8.11.4.6.4, "DVBHTMLAreaElement Interface";
- clause AC.2.14, "Object DVBHTMLAreaElement".

```
public interface DVBHTMLAreaElement
    extends DVBHTMLElement {
    public boolean isNoHRef()
    public void setNoHRef(boolean nohref)
    public String getAccessKey()
    public void setAccessKey(String accesskey)
    public String getAlt()
    public void setAlt(String alt)
    public String getHRef()
    public void setHRef(String href)
    public String getTarget()
    public void setTarget(String target)
}
```

AD.1.1.15 DVBHTMLFrameSetElement

See:

- clause 8.11.4.6.8, "DVBHTMLFrameSetElement Interface";
- clause AC.2.15, "Object DVBHTMLFrameSetElement".

```
public interface DVBHTMLFrameSetElement
    extends DVBHTMLElement {
    public String getCols()
    public void setCols(String cols)
    public String getRows()
    public void setRows(String rows)
}
```

AD.1.1.16 DVBHTMLFrameElement

See:

- clause 8.11.4.6.7, "DVBHTMLFrameElement Interface";
- clause AC.2.16, "Object DVBHTMLFrameElement".

```
public interface DVBHTMLFrameElement
    extends DVBHTMLElement {
    public Document getContentDocument()
    public String getFrameBorder()
    public void setFraneBorder(String border)
    public String getLongDesc()
    public void setLongDesc(String longdesc)
    public String getMarginHeight()
    public void setMarginHeight(String marginheight)
    public String getMarginWidth()
    public void setMarginWidth(String marginwidth)
    public String getScrolling()
    public void setScrolling(String scrolling)
    public String getSrc()
    public void setSrc(String src)
}
```

AD.1.1.17 DVBHTMLIFrameElement

See:

- clause 8.11.4.6.9, "DVBHTMLIFrameElement Interface";
- clause AC.2.17, "Object DVBHTMLIFrameElement".

```
public interface DVBHTMLIFrameElement
    extends DVBHTMLElement {
    public String getAlign()
    public void setAlign(String align)
    public Document getContentDocument()
    public String getFrameBorder()
    public void setFraneBorder(String border)
    public String getHeight()
    public void setHeight(String height)
    public String getLongDesc()
    public void setLongDesc(String longdesc)
    public String getMarginHeight()
    public void setMargineHeight(String marginheight)
    public String getMarginWidth()
    public void setMarginWidth(String marginwidth)
    public String getName()
    public void setName(String name)
    public String getScrolling()
    public void setScrolling(String scrolling)
    public String getSrc()
    public void setSrc(String src)
    public String getWidth()
    public void setWidth(String width)
}
```

AD.1.2 The org.dvb.dom.css package

All of these are in the package `org.dvb.dom.css`.

AD.1.2.1 DVBInlineStyle

This interface shall be implemented by objects that implement the `org.w3c.dom.Element` interface and which represent elements that support the style attribute.

```
public interface DVBCSSInlineStyle {
    public CSS2Properties getStyle();
};
```

See:

- clause 8.11.8.1.1, "DVBCSSInlineStyle";
- clause AC.5.1, "DVBInlineStyle".

AD.1.2.2 DVBCSSStyle

This interface shall be implemented by objects that implement the `org.w3c.dom.Element` interface that represent the root element of a document.

```
public interface DVBCSSStyle {
    public DVBCSSViewportRule getViewport();
};
```

See:

- clause 8.11.8.1.2, "DVBCSSStyle";
- clause AC.5.2, "DVBCSSStyle".

AD.1.2.3 DVBCSSViewportRule

```
public interface DVBCSSViewportRule extends org.w3c.dom.CSSRule {
    public DVBCSSViewportProperties getStyle();
};
```

See:

- clause 8.11.8.1.3, "DVBCSSViewportRule";
- clause AC.5.3, "DVBCSSViewportRule".

AD.1.2.4 DVBCSSViewportProperties

```
public interface DVBCSSViewportProperties {
    public String getPseudoClass();
    public String getArea();
    public void setArea(String area);
    public String[] getBackgroundVideoTransform();
    public void setBackgroundVideoTransform(String transform[]);
    public String[] getBackgroundVideo();
    public void setBackgroundVideo(String video[]);
    public String[] getBackgroundVideoPreserveAspect();
    public void setBackgroundVideoPreserveAspect(String aspect[]);
    public String[] getBackgroundVideoClip();
    public void setBackgroundVideoClip(String clip[]);
    public String[] getBackgroundVideoRectangle()
    public void setBackgroundVideoRectangle(String rectangle[]);
    public String getBackground();
    public void setBackground(String background);
    public String getBackgroundImageRectangle();
    public void setBackgroundImageRectangle(String rect);
    public String getInitial();
    public void setInitial(String initial);
    public int getVerticalResolution();
    public void setVerticalResolution(int res);
    public int getHorizontalResolution();
    public void setHorizontalResolution(int res);
    public String getScene();
    public void setScene(String scene);
};
```

See:

- clause 8.11.8.1.4, "DVBCSSViewportProperties";
- clause AC.5.4, "DVBCSSViewportProperties".

AD.1.3 The org.dvb.dom.environment package

All of these are in the package org.dvb.dom.environment.

AD.1.3.1 Navigator

See:

- clause 8.11.7.2.1, "Navigator Object";
- clause AC.4.1, "Navigator Object".

```
public class Navigator
    extends java.lang.Object {
    public Navigator()
    public Navigator(String appname, String appcodename,
        String appversion, String useragent)
    public String getAppCodeName()
    public String getAppName()
    public String getAppVersion()
    public String getUserAgent()
}
```

AD.1.3.2 Window

See:

- clause 8.11.7.2.2, "Window object";
- clause AC.4.2, "Window Object".

```
public class Window
    extends java.lang.Object {
    public Window()
    public Window(DVBHTMLDocument document, DVBDOMImpl impl,
        DVBHTMLCollection frames, long length, Navigator navigator,
        Window parent, Window window, Window top)
    public String getDefaultStatus()
    public void setDefaultStatus(String defaultstatus)
    public DVBHTMLDocument getDocument()
    public DVBDOMImpl getDVBDOMImpl()
    public DVBHTMLCollection getFranes()
    public long getLength()
    public Location getLocation()
    public void setLocation(Location location)
    public String getName()
    public void setName(String name)
    public Navigator getNavigator()
    public Window getParent()
    public Window getWindow()
    public Window getSelf()
    public String getStatus()
    public void setStatus(String status)
    public Window getTop()
    public void open(String url, String name, String features)
}
```

AD.1.3.3 Location

See:

- clause 8.11.7.2.3, "Location object";
- clause AC.4.3, "Location object".

```
public class Location
    extends java.lang.Object {
    public Location()
    public String getNash()
    public void setHash(String hash)
    public String getHost()
    public void setHost(String host)
    public String getHostName()
    public void setHostName(String hostname)
    public String getHRef()
    public void setHRef(String href)
    public String getPathName()
    public void setPathName(String pathname)
    public String getPort()
    public void setPort(String port)
    public String getProtocol()
    public void setProtocol(String protocol)
    public String getSearch()
    public void setSearch(String search)
}
```

AD.1.4 The org.dvb.dom.event package

All of these are in the package org.dvb.dom.event.

AD.1.4.1 DVBLifecycleEvent

See:

- clause 8.11.2.2.1, "Interface DVBLifecycleEvent";
- clause AC.3.1, "Object DVBLifecycleEvent".

```
public class DVBLifecycleEvent
    extends java.util.event {
    public DVBLifecycleEvent()
    public DVBLifecycleEvent(String typearg, boolean canbubblearg,
        boolean candisablearg, long detailarg)
    public long getDetail()
}
```

Package org.dvb.dom.bootstrap

Description

Provides the entry point to the W3C DOM APIs for Java applications.

Class Summary	
Interfaces	
`DocumentAction`	DocumentAction is used for an action that modifies a W3C Document.
`DocumentFactory`	DocumentFactory contains bootstrap methods for applications embedded in a document to access the document object model of the document in which they are contained.
`MultipleDocumentsAction`	MultipleDocumentAction is used for an action that modifies zero or more W3C Document instances.

org.dvb.dom.bootstrap DocumentAction

Declaration

```
public interface DocumentAction
```

Description

DocumentAction is used for an action that modifies a W3C Document. When an application wishes to access a Document, it creates an instance of a class that implements DocumentAction, and passes this instance to the system. The system then calls back to the user code via the DocumentAction interface.

See Also:

`MultipleDocumentsAction`, `DocumentFactory`

Methods

run(Document)

```
public void run(org.w3c.dom.Document doc)
```

Access and/or modify a W3C DOM Document. All modifications must take place during this callback. The results of retaining and using DOM references outside the context of this callback are undefined.

Parameters:

`doc` - the DOM document to modify.

org.dvb.dom.bootstrap DocumentFactory

Declaration

```
public interface DocumentFactory
```

Description

DocumentFactory contains bootstrap methods for applications embedded in a document to access the document object model of the document in which they are contained.

Since:

MHP 1.1.

Fields

DOM

```
public static final java.lang.String DOM
```

The property string to use with `XletContext.getXletProperty` in order to obtain the DocumentFactory for this Xlet (if one exists).

Methods

performAction(DocumentAction)

```
public void performAction(org.dvb.dom.bootstrap.DocumentAction act)
```

Perform an action on the document that contains the Xlet controlled by the given XletContext.

Parameters:

`act` - The action to perform. It will be called by the system either synchronously, or on a system thread.

performActionOnFrames(String[], MultipleDocumentsAction)

```
public void performActionOnFrames(java.lang.String[] names,
org.dvb.dom.bootstrap.MultipleDocumentsAction act)
```

Perform an action on a set of documents, each contained in a frame that is a part of the same application as the Xlet controlled by the given XletContext.

Parameters:

`names` - The names of the desired frames.

`act` - The action to perform. It will be called by the system either synchronously, or on a system thread.

performActionReadOnly(DocumentAction)

```
public void performActionReadOnly(org.dvb.dom.bootstrap.DocumentAction act)
```

Perform an action on the document that contains the Xlet controlled by the given XletContext. The action is called without any ability to modify the DOM.

Parameters:

`act` - The action to perform. It will be called by the system either synchronously, or on a system thread. Attempts by this action to modify the DOM shall fail.

org.dvb.dom.bootstrap MultipleDocumentsAction

Declaration

```
public interface MultipleDocumentsAction
```

Description

MultipleDocumentAction is used for an action that modifies zero or more W3C Document instances. When an application wishes to access a number of Documents simultaneously, it creates an instance of a class that implements MultipleDocumentAction, and passes this instance to the system. The system then calls back to the user code via the MultipleDocumentAction interface.

See Also:

`DocumentAction`, `DocumentFactory`

Methods

run(Document[])

```
public void run(org.w3c.dom.Document[] docs)
```

Access and/or modify a set of W3C DOM Documents. All modifications must take place during this callback. The results of retaining and using DOM references outside the context of this callback are undefined.

Parameters:

`docs` - The array of documents on which to operate. If a requested document is not found, the corresponding entry in this array will be null.

Package org.dvb.dom.inner

Description

Provides support for embedding DVB-HTML applications within the user interface of a DVB-J application.

Class Summary	
Classes	
`HTMLApplication`	Encapsulates the information needed to define a DVB-HTML application.
`HTMLInnerApplicationContainer`	Represents embedding of an DVB-HTML inner application within the user interface of a DVB-J application.

org.dvb.dom.inner HTMLApplication

Declaration

```
public class HTMLApplication extends org.dvb.application.inner.InnerApplication

java.lang.Object
  |
  +--org.dvb.application.inner.InnerApplication
        |
        +--org.dvb.dom.inner.HTMLApplication
```

Description

Encapsulates the information needed to define a DVB-HTML application.

Constructors

HTMLApplication(URL, String, String)

```
public HTMLApplication(java.net.URL physicalRoot, java.lang.String initialPathBytes,
java.lang.String parameters)
```

Constructor for instances of the class. The semantics of these shall be as defined in the main body of the present document for the equivalent information when provided through an AIT.

Parameters:

`physicalRoot` - the physical root of the application entry point.

`initialPathBytes` - a string specifying the URL path component to the entry point document. This path is relative to the root defined in the physicalRoots parameter.

`parameters` - the string that is appended to the application initial path as parameters.

HTMLApplication(URL, String, String, String[], String[])

```
public HTMLApplication(java.net.URL physicalRoot, java.lang.String initialPathBytes,
java.lang.String parameters, java.lang.String[] label, java.lang.String[] regex)
```

Constructor for instances of the class. The semantics of these shall be as defined in the main body of the present document for the equivalent information when provided through an AIT.

Parameters:

`physicalRoot` - the physical root of the application entry point.

`initialPathBytes` - a string specifying the URL path component to the entry point document. This path is relative to the root defined in the physicalRoots parameter.

`parameters` - the string that is appended to the application initial path as parameters.

`label` - an array of one or more strings specifying a label that is associated with the set of data identified by the regular expression. This label can be used for pre-fetching in a transport specific manner.

`regex` - an array of one or more strings specifying a regular expressions that can generate all URLs that are in the domain of the application.

Throws:

`java.lang.IllegalArgumentException` - if either the regex or label parameters are an array of length zero or if the lengths of these two arrays are not the same.

org.dvb.dom.inner HTMLInnerApplicationContainer

Declaration

```
public class HTMLInnerApplicationContainer extends
org.dvb.application.inner.InnerApplicationContainer

java.lang.Object
  |
  +--java.awt.Component
        |
        +--org.havi.ui.HComponent
              |
              +--org.dvb.application.inner.InnerApplicationContainer
                    |
                    +--org.dvb.dom.inner.HTMLInnerApplicationContainer
```

All Implemented Interfaces:

`org.havi.ui.HMatteLayer`, `org.havi.ui.HNavigable`, `org.havi.ui.HNavigationInputPreferred`, `java.awt.image.ImageObserver`, `java.io.Serializable`, `org.dvb.ui.TestOpacity`

Description

Represents embedding of an DVB-HTML inner application within the user interface of a DVB-J application.

Constructors

HTMLInnerApplicationContainer(HTMLApplication)

```
public HTMLInnerApplicationContainer(org.dvb.dom.inner.HTMLApplication a)
throws IOException
```

Construct an instance of this class with a DVB-HTML application as its content. The instance is initialized to present the entry point document of the application.

Parameters:

`a` - the DVB-HTML application.

Throws:

`java.io.IOException` - if an error occurred while reading the code or data for the inner application.

Methods

performAction(DocumentAction)

```
public void performAction(org.dvb.dom.bootstrap.DocumentAction act)
```

Perform an action on the DVB-HTML document.

Parameters:

`act` - The action to perform. It will be called by the system either synchronously, or on a system thread.

Annex AE:
Void

Annex AF (normative): Plug-in APIs

GEM [1], annex AF is included in the present document, with the following notes and modifications.

Support for the classes `XletContainer` and `XletSystemCall` is required in MHP 1.1 to support DVB-HTML.

Annex AG (normative): Stored application APIs

GEM [1], annex AG is included in the present document.

Annex AH (normative): Internet client APIs

GEM [1], annex AH is included in the present document.

Annex AI (normative): DVB Extensions for cryptography

GEM [1], annex AI is included in the present document.

Annex AJ (normative): Cryptographic service provider installation

GEM [1], annex AJ is included in the present document.

Annex AK (normative): Extended service selection API

GEM [1], annex AK is included in the present document.

Annex AL (normative): Extended content referencing API

GEM [1], annex AL is included in the present document with the following notes and modifications.

The class `org.dvb.locator.FrequencyLocator`, defined in this annex, is required for MHP terminals.

NOTE: In MHP, this class is used by applications that are highly dependent on a particular network. Further, the byte array used to construct an instance of this class is dependent on details of DVB SI.

Annex AM (normative): Smart card reader API

GEM [1], annex AM is included in the present document.

Annex AN (normative): Provider APIs

GEM [1], annex AN is included in the present document.

Annex AO (normative): Services and the service list

GEM [1], annex AO is included in the present document.

Annex AP (normative): Mapping between Java TV and service discovery and selection

GEM [1], annex AP is included in the present document.

Annex AQ (normative): Mapping between Java TV and broadband content guide

GEM [1], annex AQ is included in the present document.

Annex AR (normative): XML encoding for AIT

GEM [1], annex AR is included in the present document.

Annex AS (Informative): IPTV Use-cases

GEM [1], annex AS is included in the present document.

Annex AT (normative): Application Management API

GEM [1], annex AT is included in the present document.

Annex AU (normative): IPTV content referencing API

GEM [1], annex AU is included in the present document.

Annex AV (normative): Extended service list API

GEM [1], annex AV is included in the present document.

Annex AW (normative): API to DVB service discovery and selection

GEM [1], annex AW is included in the present document.

Annex AX (normative): API to DVB broadband content guide

GEM [1], annex AX is included in the present document with the following changes:

In the present document, the requirement that `DvbLocator` references be replaced does not apply.

Annex AY (normative): TV-Anytime and Java TV Integration

GEM [1], annex AY is included in the present document.

Annex AZ (normative): MHP terminal hardware API

GEM [1], annex AZ is included in the present document.

Annex BA (informative): Bibliography

- ITU-T Recommendation X.690: "Information Technology ASN.1 Encoding Rules: Specification Of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) And Distinguished Encoding Rules (DER)".

History

Document history		
V1.1.1	January 2010	Publication