# OpenCable Application Platform Specifications

## OCAP 1.1 Profile

## OC-SP-OCAP1.1-D02-080807

**DRAFT**

**Notice**

Among other things, this specification defines an optional multi-mode functionality for Host Devices. However, it does not include improvements, features, functions, corrections, and errata that were added to the OCAP 1.0.x line of specifications. A new specification that reconciles the OCAP 1.0.x specifications and the OCAP 1.1 specification was released as Draft on October 31, 2008 and is identified as OCAP 1.1.1 specification. Until OCAP 1.1.1 is released as an issued specification, this OCAP 1.1 specification may be used for general guidance on multi-mode functionality.

This document is the result of a cooperative effort undertaken at the direction of Cable Television Laboratories, Inc. for the benefit of the cable industry and its customers. This document may contain references to other documents not owned or controlled by CableLabs. Use and understanding of this document may require access to such other documents. Designing, manufacturing, distributing, using, selling, or servicing products, or providing services, based on this document may require intellectual property licenses from third parties for technology referenced in this document.

Neither CableLabs nor any member company is responsible to any party for any liability of any nature whatsoever resulting from or arising out of use or reliance upon this document, or any document referenced herein. This document is furnished on an "AS IS" basis and neither CableLabs nor its members provides any representation or warranty, express or implied, regarding the accuracy, completeness, noninfringement, or fitness for a particular purpose of this document, or any document referenced herein.

# Document Status Sheet

| | | | | |
|---|---|---|---|---|
| **Document Control Number:** | OC-SP-OCAP1.1-D02-080807 | | | |
| **Document Title:** | OCAP 1.1 Profile | | | |
| **Revision History:** | D01 – Released 11/22/2006 | | | |
| | D02 – Released 8/7/2008 | | | |
| **Date:** | August 7, 2008 | | | |
| **Status:** | ~~Work in Progress~~ | **Draft** | ~~Issued~~ | ~~Closed~~ |
| **Distribution Restrictions:** | ~~Author Only~~ | ~~CL/Member~~ | ~~CL/ Member/ Vendor~~ | **Public** |

## Key to Document Status Codes:

| | |
|---|---|
| **Work in Progress** | An incomplete document, designed to guide discussion and generate feedback that may include several alternative requirements for consideration. |
| **Draft** | A document in specification format considered largely complete, but lacking review by Members and vendors. Drafts are susceptible to substantial change during the review process. |
| **Issued** | A stable document, which has undergone rigorous member and vendor review and is suitable for product design and development, cross-vendor interoperability, and for certification testing. |
| **Closed** | A static document, reviewed, tested, validated, and closed to further engineering change requests to the specification through CableLabs. |

## Trademarks:

# Contents

# Figures



# Tables

This page left blank intentionally.

# 1 SCOPE

This section contains requirements that are extensions to [DVB-GEM 1.1] Section: 1 Scope.

This document defines a minimal profile specification for the next generation of middleware software for digital cable television set-top boxes and other digital devices to be deployed by cable operators in North America: OpenCable Application Platform 1.1 (OCAP 1.1).

The OCAP 1.1 Profile is based on the [DVB-GEM 1.1] and [DVB-MHP1.1.2] specifications. Each section of the OCAP 1.1 Profile contains a correspondence Table, indicating the corresponding [DVB-GEM 1.1] or [DVB-MHP1.1.2] section and how OCAP 1.1 complies with [DVB-GEM 1.1] or [DVB-MHP1.1.2] section or deviates from the [DVB-MHP1.1.2] sections. Whenever appropriate, OCAP 1.1 directly references [DVB-GEM 1.1] or [DVB-MHP1.1.2].

## 1.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

This section corresponds to [DVB-MHP1.1.2] (there is no corresponding section in [DVB-GEM 1.1]) as follows:

***Table 1–1 - Correlation between OCAP 1.1, [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] and [DVB-MHP1.1.2] Sections | GEM and MHP Compliance |
|---|---|---|
| 1 Scope | 1 Scope | Extension |
| 1.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension |
| 1.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension |
| No Corresponding Section | 1 Scope | Compliance |

## 1.2 OCAP 1.1 Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] section.

### 1.2.1 OCAP 1.1 Purpose

OCAP 1.1 is an application interface that includes all required Application Program Interfaces (APIs), content and data formats, and protocols up to the application level. Applications developed to the OCAP 1.1 Profile will be executed and/or displayed on OpenCable certified retail host devices. The OCAP 1.1 Profile will allow cable operators, content providers, and CD manufacturers to deploy applications and services on all OpenCable-compliant host devices.

The OCAP 1.1 implementation SHALL be applicable to a wide variety of hardware and operating systems to allow Consumer Electronics (CE) manufacturers flexibility in implementation. A primary objective in defining OCAP 1.1 is to enable competing implementations of the OCAP 1.1 interface for selection by the CE manufacturers.

### 1.2.2 OCAP 1.1 Application Areas

#### *1.2.2.1 Application Classification (Normative)*

The following information is normative for the OCAP 1.1 Profile. It is specific to OCAP 1.1 and is not contained within [DVB-MHP1.1.2].

This specification has identified three classifications of applications which can run on an OCAP 1.1 terminal:

- bound applications
- unbound applications
- non-cable applications

##### *1.2.2.1.1 Bound Applications*

Bound applications are those applications which are bound to, or associated with, the currently tuned channel. When the viewer tunes away from the current channel, the bound application goes out of scope and SHALL be terminated unless it is also associated with the newly tuned channel. An application which is associated with the channel before and the channel after a tuning operation SHALL continue running uninterrupted after the tuning operation if it was running before the tuning occurred.

##### *1.2.2.1.2 Unbound Applications*

Unbound applications are not tied to any specific channel. They are free to run across channels and do not go out of scope when the viewer tunes away from any specific service.

##### *1.2.2.1.3 Non-cable Applications*

OpenCable recognizes the need for applications placed on a particular OCAP 1.1 terminal by means outside the scope of the OCAP specification - e.g., by the manufacturer. These applications may be written for a specific host and, if so, are not guaranteed to run across different hardware or software (that is, OS) configurations.

#### *1.2.2.2 Application Functional Overview (Informative)*

The following information is informative to the OCAP 1.1 Profile. It attempts to clarify issues in the [DVB-MHP1.1.2], as well as provide additional information to the normative requirements.

This section identifies the applications and services that could be made available to the user, or viewer, when using an OCAP 1.1-compliant terminal. These applications will typically be implemented on top of OCAP 1.1 as bound or unbound applications.

The descriptions of the applications are intended to demonstrate the scope of services required from OCAP 1.1.

##### *1.2.2.2.1 Electronic Program Guide / Interactive Program Guide*

The electronic program guide (EPG) application (also known as enhanced program guide, or interactive program guide) is the interface for selecting traditional television channels, as well as selecting some advanced services. It also may contain windowed video displaying ads or clips from the highlighted program. The cable operator determines its exact design since the cable operator downloads it to the host device.

In the absence of authorization from the cable operator, a native guide may exist that displays information about those events carried on the network "in the clear." This allows a host device connected to the cable operator's network to receive the unencrypted programming that contains embedded service information without a CableCARD device.

1.2.2.2.1.1 Current Programming

Current programming displays what is being offered as a service by the cable operator. This has been accomplished traditionally through a user interface using a grid of times and channels. The presentation also includes other events available from the cable operator. These include pay-per-view (PPV), near-video-on-demand (NVOD), and video-on-demand (VOD). Any advanced services, such as email, may also be displayed in the current programming section of an EPG.

1.2.2.2.1.2 Current Purchases

When the user has purchased an event, such as VOD, that event is also displayed as part of the EPG so the user can access it easily. The event will appear for as long as the purchase is valid. In other cases, such as premium channels, the event will appear as part of the EPG and act as a kind of advertisement although it has not been purchased and is not currently authorized to the user.

1.2.2.2.1.3 Local Archive

If events can be stored locally for future viewing (as described in Section 1.2.2.2.3, those events would also be displayed as part of the guide.

#### *1.2.2.2.2 Watching TV*

While not actually considered an application, some aspects of viewing the video and audio do reflect capabilities of the OpenCable Application Platform.

1.2.2.2.2.1 SDTV

OCAP 1.1 provides the presentation of the standard definition television signal under control of an application.

1.2.2.2.2.2 HDTV

If high definition television (HDTV) is being received, the user can set various options to control its presentation. Factors such as aspect ratios and resolutions are selected by an application based upon what is supplied in the received signal, what the host device hardware supports, and what the user prefers.

1.2.2.2.2.3 Audio

The received signal may include multiple audio tracks. These might contain stereo, surround, and alternate languages. The signal might also contain closed caption text for the audio. The user's preferences SHALL determine which of these sources is chosen.

1.2.2.2.2.4 Picture-in-Picture

If the host device hardware supports this feature, multiple video streams may be received, decoded, and displayed. Picture-in-picture is the current implementation of this feature. More than two video pipelines could be available on an OCAP 1.1 terminal and is supported by the OpenCable Application Platform.

1.2.2.2.2.5 Time-shifting (Digital Recording / Playback)

This technology allows an audio/video stream to be digitally recorded and replayed at a later time (for example, PVR). OCAP 1.1 terminals may be built which include the digital recording and playback technology. An application may be designed to provide features that are common to digital video recorders/players.

1.2.2.2.2.6 Emergency Alert System

The Federally mandated Emergency Alert System (EAS) SHALL be implemented. It is up to the cable operator's policy as to whether the alert is displayed as an overlay or as a forced tuning to a specific channel to present the information. The OpenCable Application Platform supports the capability to modify attributes of resident EAS.

1.2.2.2.2.7 Content Advisory / Parental Control

The content advisory descriptor or "V chip" VBI information allows the user to limit the material presented to the viewer. The OpenCable Application Platform supports content advisory/parental control features through a flexible and expressive exception mechanism.

#### *1.2.2.2.3 Pay-Per-View*

A pay-per-view (PPV) application allows the user to purchase an event in advance and then view it at its scheduled time. Boxing events are typically sold this way today. This application simply interacts with the purchase system supplied by the cable operator to make the purchase. The conditional access system provided in the CableCARD device handles decryption of the signal when the event is viewed. The EPG may provide some sort of reminder of a scheduled purchase when it is time for the event.

#### *1.2.2.2.4 Near-Video-On-Demand*

A near-video-on-demand (NVOD) application allows the user to start viewing an event very shortly after purchasing it. A NVOD application actually represents a group of events that contain the same audio/video stream offset by a small amount of time. The user is directed to the stream that will begin next. The user can jump around the event in increments of the time offset to simulate fast forward and rewind.

#### *1.2.2.2.5 Video-On-Demand*

A video-on-demand (VOD) application actually has two parts or phases. The first phase will present the user with a catalogue of choices. The second phase controls the stream selected from the catalogue.

A VOD application is most likely a bound application because the video is being broadcast on a specific channel.

1.2.2.2.5.1 Catalog

There are many possibilities for presenting the catalogue to the viewer. The catalogue software will be downloaded from the cable operator and will be specific to that operator. This allows the operator the maximum flexibility and control over the interface.

1.2.2.2.5.2 Control

The control portion of the VOD application will also be downloaded. It will probably use some variation of the DSM-CC UU stream control, but could be completely unique to the cable operator.

#### *1.2.2.2.6 Email*

An email application will allow users to send and receive email using their television. An email application will probably use SMTP, POP, or IMAP, as those are the most used Internet standards for email. The cable operator is free to download a specialized client for any email protocol desired on its network.

#### *1.2.2.2.7 Chat / Conferencing*

A chat application will allow a user to chat with another viewer. In this case, the users will use their televisions to send and receive instant messages. The chat client and directory supported by the cable operator determines what is

downloaded to the host device. If the cable operator allows, and the user prefers, an invitation to chat might appear floating over the video while the user is watching TV. The user can then choose to initiate chat or refuse chat. The chat window could appear floating over the video and allow the user to continue to watch and chat at the same time, perhaps discussing the program with the other chat participants.

A chat/conferencing application would most likely be an unbound application. You should be able to continue chatting even if you decide that you want to tune to a different channel.

#### *1.2.2.2.8 IP Telephony*

IP Telephony involves delay/jitter-sensitive audio and possibly video streams. It is possible to connect standard telephones to the host device and allow the user to receive and place calls through the cable operator's network if the cable operator allows such services.

#### *1.2.2.2.9 Games*

Some games will be implemented using OCAP 1.1. OCAP 1.1 supports game applications by providing APIs for playing sound effects, animation, and windowed video. To support multi-player network games, the application will need to provide user-to-user and server-based messaging.

Some games will require response times faster than can be achieved using OCAP 1.1. "Twitch" games are an example. In this case, the application may have to use the native-code interface to specific OCAP 1.1 terminals.

#### *1.2.2.2.10 Music / Radio*

Music broadcast in MPEG transport streams is another service potentially offered by the cable operators. OCAP 1.1 supports this by providing APIs for decoding audio without video. Audio decoders beyond the native audio format are supported by the OpenCable Application Platform through downloadable software decoders.

Music/radio applications are most likely bound applications because they are tied to the currently tuned channel. For example, if you are listening to a country music channel, the music would stop when you tuned to a new channel.

#### *1.2.2.2.11 E-commerce*

E-commerce allows you to exchange goods and services over the network, usually with some kind of monetary transaction involved. Examples of e-commerce can include making travel arrangements electronically, buying computer hardware or software on-line, purchasing tickets for an upcoming concert or event, etc.

Security is the key concern for applications in this area. The application SHALL provide excellent security for transactions over the network and data stored locally.

##### 1.2.2.2.11.1 Shopping

Presentation of attractive interactive catalogues is paramount for a shopping application to succeed. Full-screen or windowed video and flashy transitions or other special effects will find application here. It also needs to be very clear when a purchase is being made, for what amount, and from what account. User identification and transaction security needs to be assured. Privacy concerns about shopping habits will be addressed by the cable operator's policies.

##### 1.2.2.2.11.2 Banking

The interface for a banking application is not as flashy as shopping, but the need for user identification and transaction security is higher. The application needs to support user identification and excellent transaction security.

A banking application is most likely an unbound application because it would not need to be tied to any specific channel. For example, if you decide to check the balance of your savings account while watching a basketball game, you should be able to do so.

# 2 REFERENCES

This section contains requirements that are extensions to [DVB-GEM 1.1] Section: 2 References and are extensions of [DVB-MHP1.1.2] Section: 2 References.

## 2.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that does not correspond to any [DVB-GEM 1.1] Section.

***Table 2–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 2 References | 2 References | Extension | 2 References | Extension |
| 2.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 2.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 2.2.1 Normative References | Section 2, References | Extension | Section 2, References | Extension |
| 2.2.2 Informative References | Annex C | Extension | Annex C | Extension |

**Note**: OCAP 1.1 indices in Table 2–2 do not have a one-to-one relationship with corresponding indices used in [DVB-MHP1.1.2].

## 2.2 OCAP 1.1 Specific Requirements

This subsection contains OCAP-specific requirements that does not correspond to any [DVB-GEM 1.1] Section.

### 2.2.1 Normative References

The following information is normative for the OCAP 1.1 Specification. It is specific to OCAP 1.1 and is not contained within [DVB-MHP1.1.2].

**Note:** Information contained in these normative references is required for all implementations. Notwithstanding, intellectual property rights may be required to use or implement these normative references.

#### *2.2.1.1 List of Normative References for OCAP 1.1*

The following documents contain provisions which, through reference in this text, constitute provisions of the present documents. References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific:

- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

A non-specific reference to an ETS SHALL also be taken to refer to later versions published as an EN with the same number.

Some known errata in these references are identified in Annex A of [DVB-MHP1.1.2]. These errata take precedence over the published reference.

#### *2.2.1.1.1 Notes*

The following sections apply to particular sources of documents in Table 2–2 Note column:

1. Where the reference is to an ISO specification, it is considered to be a non-specific reference. Additionally, officially published amendments and corrigenda are considered to automatically update the referenced document.
2. Where an ISBN number is provided for a referenced document, it is considered to be a specific reference.
3. References to RFCs are considered to be specific references. An RFC being indicated obsoleted by another RFC is not considered significant.
4. URL references with note [4] are provided for convenience to access the document in electronic form.
5. URL references with note [5] are the normative method to access the information.
6. ETSI specifications are available from the ETSI server at: http:// www.etsi.org. However, the ETSI server provides the current edition of the specification and in every case, this specification makes specific references which in the future may not be the current reference.
7. Available under NDA from CableLabs.
8. ITU recommendations are available from ITU at http://www.itu.int for a fee.
9. ATSC specifications are available from http://www.atsc.org.
10. SCTE specifications are available from http://www.scte.org.
11. CableLabs specifications are available from: http://www.opencable.com/specifications/
12. Nielsen Media Research reports are available to the public upon individual request. Please contact them directly at info@nielsenmedia.com or by telephone at 1-646-654-8354.

***Table 2–2 - OCAP 1.1 Normative References***

| Spec | Description | See Notes Section 2.2.1.1.1 | MHP Index |
|---|---|---|---|
| [ATSC A/52B] | ATSC Digital Audio Compression Standard (AC-3), ATSC A/52B, 11 June 2005. | [9] | NA |
| [ATSC A/53E] | ATSC Digital Television Standard, ATSC A/53E, 23 December 2005. | [9] | NA |
| [ATSC A/96] | ATSC Interaction Channel Protocols, ATSC A/96, 3 February 2004 | [9] | NA |
| [CCCP 2.0] | OpenCable CableCARD Copy Protection System 2.0 Interface Specification, OC-SP-CCCP2.0-I04-060803, August 3, 2006, Cable Television Laboratories, Inc. | [11] | NA |
| [CCIF 2.0] | OpenCable CableCARD Interface 2.0 Specification, OC-SP-CCIF2.0-I08-061031, October 31, 2006, Cable Television Laboratories, Inc. | [11] | NA |
| [CEA-766] | U.S. and Canadian Rating Region Table (RRT) and Content Advisory Descriptor for Transport of Content Advisory Information Using ATSC A/65A Program and System Information Protocol (PSIP), CEA-766-B, July 31, 2006. | | NA |
| [CHILA] | CableCARD-Host Interface License Agreement, CHILA http://www.opencable.com/downloads/CHILA.pdf | [7] | NA |

| Spec | Description | See Notes Section 2.2.1.1.1 | MHP Index |
|---|---|---|---|
| [CORBA/ IIOP] | The Common Object Request Broker: Architecture and Specification, CORBA/IIOP 2.1, Object Management Group. http://www.sei.cmu.edu/str/descriptions/corba.html | [5] | [2] |
| [DAVIC] | DAVIC 1.4.1 Specification Part 9, DAVIC 1.4.1p9., June 1999. http://www.mhp.org/mhp_technology/mhp_1_0/references_in_mhp_1_0/ | [5] | [3] |
| [DVB-GEM 1.1] | DVB Globally Executable MHP 1.1, DVB-GEM 1.1.2, Draft ETSI TS 102 543 V1.1.1, July 2006 http://www.mhp.org/mhp_technology/gem/a103.tm3567.GEM1.1.pdf<br>Note: normative references [1] and [7] in GEM 1.1 should refer to [DVB-MHP1.1.2] and [DVB-MHP Errata #1] and will be revised through the DVB-MUG Corrigenda process. | | |
| [DVB-MHP1.1.2] | DVB Multimedia Home Platform (MHP) Specification 1.1.2.<br>http://www.mhp.org/mhp_technology/mhp_1_1/mhp_a0068r1.zip<br>Note: reference 119 in this specification refers to ETSI specification TS 102 823. | | |
| [DVB-MHP Errata #1] | MHP Specification Version 1.1.2 Errata #1 (tm 3570/Tam0941r2), June 2006.<br>http://www.mhp.org/mhp_technology/mhp_1_1/tm3570.tam0941r2.MHP1.1.2_errata1.clean.pdf | | |
| [DVB-SAD] | Digital Video Broadcasting (DVB): Specification for the carriage of synchronized auxiliary data in DVB transport streams, ETSI TS 102 823 v1.1.1 (2005-11) http://pda.etsi.org/pda/queryform.asp. | | |
| [DVS-766] | Stream Conditioning for Addressable Ad Insertion. | | |
| [EIA-608-B] | EIA Standard Line 21 Data Services, EIA/CEA 608-B, October 2000. http://global.ihs.com/ | | NA |
| [EIA-708-B] | Digital Television (DTV) Closed Captioning, EIA-708-B, 12/29/1999. | | |
| [EN 301 192] | Digital Video Broadcasting (DVB); Specification for Data Broadcast, EN 301 192, 1.4.1 (2004-11). | [6] | [5] |
| [ETS 300 802] | Digital Video Broadcasting (DVB); Network Independent Protocols for Interactive Services, ETSI 300 802, 1 November 1997. | [6] | [17] |
| [ETSI EN 300 468] | Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB Systems, ETSI EN 300 468, v1.5.1, 2003-05. | [6] | |
| [ETSI TR 101 154] | Digital Video Broadcasting (DVB); Implementation guidelines for the use of MPEG-2 systems; Video and audio in satellite, cable and terrestrial broadcasting applications, ETSI TR 101 154, version V1.7.1 (2005-06). | | |
| [ETSI TR 101 202] | Digital Video Broadcasting (DVB); Guidelines for the use of EN 301 192, ETSI TR 101 202, 1.2.1 (2003-01). | | [50] |
| [FCC 98-36] | Report and Order, Technical Requirements to Enable Blocking of Video Programming based on Program Ratings Implementation of Sections 551(c), (d), and (e) of the Telecommunications Act of 1996, FCC 98-36, ET Docket No. 97-206, Adopted March 12, 1998; Released March 13, 1998. | | |

| Spec | Description | See Notes Section 2.2.1.1.1 | MHP Index |
|---|---|---|---|
| [HAVi] | HAVi Level 2 User Interface specification, HAVi 1.1 http://www.havi.org/ | | [51] |
| [HOST 2.0] | OpenCable Host Device Core Functional Requirements, OC-SP-HOST2.0-CFR-I11-061031, October 31, 2006, Cable Television Laboratories, Inc. | [11] | NA |
| [ISO 10646] | Information technology—Universal multiple-octet coded character set (UCS), ISO 10646; 2003, December 2003.<br>Amendment 1, November 2005 | [1] | [17] |
| [ISO 11172-2 | Information Technology-Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbits/s-part 2: video, 2003. | [1] | |
| [ISO 11172-3] | Information technology-coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbits/s-Part 3: Audio (Note: known as MPEG-1 Audio), ISO/IEC 11172-3, August 1993. | [1] | [21] |
| [ISO 13818-1] | Information technology—Generic coding of moving pictures and associated audio information: Systems, ISO/IEC 13818-1:2005, October 2005. | [1] | [22] |
| [ISO 13818-2] | Information technology—Generic coding of moving pictures and associated audio information-Part 2: Video, ISO/IEC 13818-2, December 2000. | [1] | [23] |
| [ISO 13818-3] | Information Technology - Generic coding of moving pictures and Associated Audio Information - Part 3: Audio. | [1] | |
| [ISO 13818-6] | Information technology—Generic coding of moving pictures and associated audio information: Extensions for DSM-CC, ISO/IEC 13818-6, September 1998. | [1] | [25] |
| [H.222.1] | ITU-T Rec. H.222.1 (03/96), Multimedia multiplex and synchronization for audiovisual communication in ATM environments. | [8] | |
| [Java RMI] | Java Remote Method Invocation Specification, Java RMI, Rev 1.8, Java 2 SDK Std Ed, v1.4. http://java.sun.com/products/jdk/rmi/ | | |
| [Java TV] | Java TV API Specification, Java TV, Version 1.0. http://java.sun.com/products/javatv/ | | [52] |
| [JFIF] | JPEG File Interchange Format, Eric Hamilton, C-Cube Microsystems, JFIF. http://www.w3.org/Graphics/JPEG/jfif3.pdf | [5] | [36] |
| [JMF] | Sun Microsystems Java Media Framework Specification, Java Media Framework (JMF), Version 2.1.1e, May 2003. http://java.sun.com/products/java-media/jmf/index.jsp | [5] | [33] |
| [JNI] | Java Native Interface Specification, JNI. http://java.sun.com/products/jdk/1.2/docs/guide/jni/spec/jniTOC.doc.html | | NA |
| [JSSE] | Java Secure Sockets Extension API Specification, JSSE, Version 1.0.2 http://java.sun.com/ products/jsse/doc/apidoc/index.html | | [61] |

| Spec | Description | See Notes Section 2.2.1.1.1 | MHP Index |
|---|---|---|---|
| [JVM Err] | Errata for the Virtual Machine Specification (1st Edition), http://java.sun.com/docs/books/vmspec/errata.html JVM Errata<br>Part of ISBN:1-892488-25-6 | [2]<br>[4} | [68] |
| [Metrics] | OpenCable Receiver Metrics Gathering Specification, OC-SP-Metrics-I01-061229, December 29, 2006, Cable Television Laboratories, Inc. | | |
| [OC-SEC] | OpenCable System Security Specification, OC-SP-SEC-I07-061031, October 31, 2006. | [11] | NA |
| [PBP1.1] | Personal Basis Profile 1.1 for the J2ME Platform (JSR 217). http://www.jcp.org/ | [5] | |
| [PNG] | Portable Network Graphics, PNG, November 2003. http://www.w3.org/TR/PNG/ | [5] | [38] |
| [RFC 1738] | Uniform Resource Locators (URL), IETF RFC 1738 OBSOLETED BY 2 RFCs, see comment sheet, December 1994. | [3] | NA |
| [RFC 2396] | IETF Uniform Resource identifiers (URI): Generic Syntax, IETF RFC 2396, August 1998. | [3] | [42] |
| [RFC 793] | (TCP) Transmission Control Protocol, J. Postel, IETF RFC 793/STD0007, Sept 1, 1981. | [3] | [45] |
| [SCTE 07] | ANSI/SCTE 07 2006, Digital Video Transmission Standard for Cable Television. | [10] | NA |
| [SCTE 18] | SCTE 18 2002, Emergency Alert Message for Cable, (ANSI-J-STD-042-2002). | [10] | NA |
| [SCTE 19] | SCTE 19-2006, Standard Methods for Isochronous Data Services Transport. | [10] | NA |
| [SCTE 20] | ANSI/SCTE 20 2004, Methods for Carriage of Closed Captions and Non-Real Time Sampled Video. | [10] | NA |
| [SCTE 21] | ANSI/SCTE 21 2001, R2006, Standard for Carriage of NTSC VBI Data in Cable Digital Transport Streams. | [10] | NA |
| [SCTE 26] | ANSI/SCTE 26 2004, Home Digital Network Interface Specification with Copy Protection. | [10] | NA |
| [SCTE 40] | ANSI/SCTE 40 2004, Digital Cable Network Interface Standard. | [10] | NA |
| [SCTE 43] | ANSI/SCTE 43 2005, Digital Video Systems Characteristics, Standard for Cable Television. | [10] | NA |
| [SCTE 53] | ANSI/SCTE 53 2002, Methods for Asynchronous Data Transport. | [10] | NA |
| [SCTE 54] | ANSI/SCTE 54 2004, Digital Video Service Multiplex and Transport System Standard for Cable Television. | [10] | NA |
| [SCTE 55-1] | ANSI/SCTE 55-1 2002, Digital Broadband Delivery System: Out of Band Transport—Mode A. | [10] | NA |
| [SCTE 55-2] | ANSI/SCTE 55-2 2002, Digital Broadband Delivery System: Out of Band Transport—Mode B. | [10] | NA |
| [SCTE 57] | ANSI/SCTE 57 2003, System Information for Satellite Distribution of Digital Television for Cable and MMDS. | [10] | NA |

| Spec | Description | See Notes Section 2.2.1.1.1 | MHP Index |
|---|---|---|---|
| [SCTE 65] | ANSI/SCTE 65 2002, Service Information Delivered Out-Of-Band for Digital Cable Television. | [10] | NA |

### 2.2.2 Informative References

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: Annex C and are extensions of [DVB-MHP1.1.2] Section: Annex C.

The [DVB-MHP1.1.2] reference Table should be replaced with the following:

***Table 2–3 - OCAP 1.1 Informative References***

| Spec | Description | See Notes Section 2.2.1.1.1 |
|---|---|---|
| [ACN] | Summary of AMOL I and AMOL II Specifications, ACN 403-1218-024, Nielsen Media Research. | [12] |
| [CEA-679-B] | National Renewable Security Standard (NRSS), CEA-679-B, October 2000. | |
| [DSG] | DOCSIS Set-Top Gateway (DSG) Interface Specification, CM-SP-DSG-I09-061222, Cable Television Laboratories, Inc. | |
| [EIA 516] | Joint EIA/CVCC Recommended Practice for Teletext: North American Basic Teletext Specification (NABTS), CEA 516, May 1988. | NA |
| [EIA-775-A] | DTV 1394 Interface Specification, EIA-775-A, April 1, 2000. | NA |
| [FCC-CC] | FCC rules for closed captioning. http://ftp.fcc.gov/cgb/dro/caption.html | |
| [FCC-EAS] | FCC rules for emergency alert system. http://www.fcc.gov/eb/eas/ | |
| [JSR] | Application Isolation API Specification, Java Specification Request, JSR 121. http://www.jcp.org/en/jsr/detail?id=121 | |
| [JVM Tech] | JVM Technology. http://java.sun.com/products/jdk/1.2/docs/guide/misc/threadPrimitiveDeprecation.html | |

# 3 GLOSSARY

This section contains requirements that are extensions to [DVB-GEM 1.1] Section: 3 Definitions and abbreviations and are extensions of [DVB-MHP1.1.2] Section: 3 Definitions and abbreviations.

## 3.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that does not correspond to any [DVB-GEM 1.1] Section. Section 3. This section corresponds to [DVB-MHP1.1.2] and [DVB-GEM 1.1] as follows:

***Table 3–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 3 Glossary | 3 Definitions and abbreviations | Extension | 3 Definitions and abbreviations | Extension |
| 3.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 3.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 3.2.1 Definitions | 3.1 Definitions | Extension | 3.1 Definitions | Extension |

## 3.2 OCAP 1.1 Specific Requirements

This subsection is an OCAP specific section that does not correspond to any [DVB-GEM 1.1] Section.

### 3.2.1 Definitions

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 3.1 Definitions and are extensions of [DVB-MHP1.1.2] Section: 3.1 Definitions.

The following definitions are used in this specification:

**Abstract service** — A mechanism to group a set of related unbound applications where some aggregator has taken the responsibility to ensure that the set of related applications work together. This is a generalization of a broadcast service to support applications not related to any broadcast TV service. A set of resident applications which an MSO has packaged together (e.g., chat, email, WWW browser) could comprise one abstract service.

**Abstract Windowing Toolkit (AWT)** — A Java package that supports graphical user interface (GUI) programming

**Advanced Television System Committee (ATSC)** — An international organization of 200 members that is establishing voluntary technical standards for advanced television systems.

**AIT** — See Application Information Table (AIT).

**API** — See Application Program Interface (API).

| | |
|---|---|
| **Application** | An application is a functional implementation realized as software running in one or spread over several interplaying hardware entities. |
| **Application boundary** | A concise general description of the data elements (code files, images, etc.) used to form one application and the logical locator of the entry point, the application boundary is described by a regular expression over the URL language. Where no such boundary is drawn, the default boundary SHALL be the entire set of documents that the OpenCable platform can access. |
| **Application Information Table (AIT)** | Provides information about the activation state of service bound applications. |
| **Application manager** | The application manager is the entity in the OpenCable Application Platform that is responsible for managing the lifecycle of the applications. It manages both the bound and unbound applications. |
| **Application platform** | An application platform is the collection of application program interfaces and protocols on which content and applications are developed. |
| **Application Program Interface (API)** | An Application Program Interface is the software interface to system services or software libraries. An API can consist of classes, function calls, subroutine calls, descriptive tags, etc. |
| **Aspect ratio** | The aspect ratio refers to the ratio of width to height of a picture. Standard definition television uses a 4:3 aspect ratio. High definition television uses a 16:9 aspect ratio. |
| **ATSC** | See Advanced Television System Committee (ATSC). |
| **AWT** | See Abstract Windowing Toolkit (AWT). |
| **background applications** | Applications from one environment which are running when that environment is not the selected environment and which can neither display graphics on a display used by the selected environment nor receive input focus. The terms "background application" and "application running in background mode" are equivalent |
| **Backus Naur Form (BNF)** | A formal notation used to define the syntax of a language. |
| **BFS** | See Broadcast File System (BFS). |
| **Boot loader** | The boot loader is a software component, provided with the host device, which is responsible for loading a cable loader or loading a software stack from non-cable proprietary I/O. |
| **BNF** | See Backus Naur Form (BNF). |
| **Broadcast** | A broadcast is a service that is delivered to all customers. Each customer may select a particular broadcast channel out of many. |
| **Broadcast application** | A broadcast application is an application running on the set-top converter that is loaded through in-band information, inserted either at the headend or by a content provider farther upstream. |

| | |
|---|---|
| **Broadcast File System (BFS)** | A broadcast file system is a data carousel system by which application data can be stored on an application server and transmitted frequently to the set-top converters for application use. |
| **Bound application** | Bound applications are those applications which are bound to, or associated with, a particular service made available by the cable operator. |
| **CA** | See Conditional access (CA) and encryption. |
| **Cable application** | Application deployed to the iDCR by means inside the scope of OCAP – both unbound applications from MSOs and bound applications from content providers. |
| **CableCARD** | A CableCARD device, also referred to as "Point of Deployment" (POD), is a detachable device distributed by cable providers, that connects to the home receiver. The interface between the CableCARD device and the receiver is specified by the OpenCable platform. CableCARD functionality includes copy protection and signal demodulation. |
| **cable environment** | The technologies, rules and policies provided by the OCAP specification and the monitor application |
| **Cable operator** | See Multiple Systems Operator (MSO) |
| **Carousel** | See Object carousel. |
| **CATV** | An abbreviation for "Cable TV". |
| **CCI** | See Copy Control Information (CCI). |
| **Certificate Revocation List (CRL)** | A list of revoked certificates published by each certificate authority. |
| **CHILA** | CableCARD Host Interface License Agreement. |
| **Conditional access (CA) and encryption** | Conditional access and encryption is a system that provides selective access to programming to individual customers in exchange for payment. |
| **Content** | Content is typically used to refer to audio, video, and graphic materials used by a service. Sometimes data and applications are also grouped into this term. |
| **Content protection/copy protection (CP)** | Content protection is a mechanism to protect the unauthorized copying of video and audio programming. |
| **Coordinated Universal Time (UTC)** | A reference time standard established by the CCIR (a predecessor of the ITU) and maintained by the Bureau International des Poids et Mesures (BIPM). Formerly Greenwich Mean Time (GMT). |
| **Copy Control Information (CCI)** | This information is stored on a CableCARD device and delivered to the host to control the copying of content. It is delivered to the CableCARD device from the headend, based on arrangements made between the network operator and the content provider. |
| **Corrigenda** | Errata. |

| | |
|---|---|
| **CP** | See Content protection/copy protection (CP). |
| **CRC** | See Cyclic Redundancy Check (CRC). |
| **CRL** | See Certificate Revocation List (CRL). |
| **cross-environment applications** | Applications from one environment able to run when another environment is selected. The terms "cross-environment application" and "application running in cross-environment mode" are equivalent. Cross environment applications are the union of cross-service applications and utility applications. |
| **Cross-service applications** | Those cross-environment applications which provide a service to the end-user. e.g., instant messaging or caller-id. |
| **Cyclic Redundancy Check (CRC)** | An algorithm to detect data corruption. |
| **Data-Over-Cable Service Interface Specifications (DOCSIS )** | DOCSIS is a suite of specifications for retail cable modems. |
| **DAVIC** | See Digital Audio Visual Council (DAVIC). |
| **DECT** | See Digital Enhanced Cordless Telecommunications (DECT) |
| **DHCP** | See Dynamic host configuration protocol (DHCP) |
| **Digital Audio Visual Council (DAVIC)** | DAVIC is an international consortium working on the development of standards for interactive television. |
| **Digital Enhanced Cordless Telecommunications (DECT)** | A European standard governing pan-European digital mobile telephony. Specified in DVB MHP as a type of return channel network interface for use in receiving and transmitting IP packets. |
| **Digital Program Insertion (DPI)** | Insertion of alternative content into digitally encoded content in response to messaging in the stream. |
| **Digital Storage Media-command and Control (DSM-CC)** | A syntax defined in the MPEG-2 standard, part 6, for VCR like control over a bitstream. Playback commands include Still-Frame, Fast-Forward, Advance, Goto. |
| **Digital Video Broadcasting (DVB)** | Digital video broadcasting is a European standard for digital television. |
| **Digital Video Subcommittee (DVS)** | An ANSI-sponsored standardization subcommittee of the SCTE. |
| **DII** | See DownloadInfoIndication (DII) |

| | |
|---|---|
| **Direct request** | An explicit request by the end-user for the iDCR to perform some operation. An explicit request by the end-user to perform an operation immediately is always a direct request. An explicit request by the end user to perform some operation at a later time is always a direct request. A generalized opt-in by the end-user is not a direct request. |
| **DOCSIS** | See Data-Over-Cable Service Interface Specifications (DOCSIS ) |
| **Document Type Definition (DTD)** | A formal grammar to specify the structure and permissible values of XML documents. |
| **Dolby AC-3** | Dolby AC-3 refers to the audio encoding format adopted by the ATSC for its advanced television audio encoding. Also known as Dolby digital. |
| **Domain of an application** | The domain of an Xlet characterizes the space within which the Xlet is able to execute. This includes both the connection where the Xlet is delivered and other connections where an already executing Xlet is allowed to continue executing.<br><br>An application cannot run outside its domain. The maximum lifetime of an application extends from the moment the user navigates to its domain until the moment that the user navigates away from its domain.<br><br>In the broadcast case, a connection corresponds to a DVB-service. Broadcast signaling indicates which services can load an application and which services allow an already active application to continue. |
| **DownloadInfoIndication (DII)** | A message that signals the modules that are part of a DSM-CC object carousel. |
| **DSM-CC** | See Digital Storage Media-command and Control (DSM-CC). |
| **DTD** | See Document Type Definition (DTD). |
| **DVB** | See Digital Video Broadcasting (DVB). |
| **DVB network** | A DVB-network is a collection of MPEG-2 Transport Stream multiplexes transmitted on a single delivery system. For example, all digital channels on a specific cable system make up a DVB network. |
| **DVB-J** | DVB-J refers to the Java platform as defined as part of the [DVB-MHP1.1.2]. For the OCAP 1.1 implementation, DVB-J is part of the Execution Engine. |
| **DVB-J API** | DVB-J API refers to one of the Java APIs standardized as part of the [DVB-MHP1.1.2]. For the OCAP 1.1 implementation, the DVB-J APIs are supported in the Execution Engine. |
| **DVB-J application** | A DVB-J application is a set of DVB-J classes that operate together and need to be signaled as a single instance to the application manager so that it is aware of its existence and can control its lifetime through a lifecycle interface. DVB-J applications as specified by the [DVB-MHP1.1.2] are not directly supported by OCAP 1.1 without modifications pertaining to this specification. |
| **DVS** | See Digital Video Subcommittee (DVS). |

| | |
|---|---|
| **Dynamic host configuration protocol (DHCP)** | The DHCP is an Internet standard for assigning IP addresses dynamically to IP hosts. |
| **EAS** | See Emergency Alert System (EAS). |
| **EE** | See Execution Engine (EE). |
| **EIA** | See Electronic Industry Alliance (EIA). |
| **Electronic Industry Alliance (EIA)** | An industry association accredited by ANSI (American National Standards Institute) to develop standards in the areas of electronic components, consumer electronics, electronic information, and telecommunications. |
| **Electronic Program Guide (EPG)** | An electronic program guide is an application that displays television program information, including program name, start time, and duration. |
| **Elementary Stream (ES)** | An elementary stream is a generic term for one of the coded video, coded audio, or other coded bit streams. One elementary stream is carried in a sequence of PES packets with one and only one stream_id. |
| **Emergency Alert System (EAS)** | The US Federal system for alerting the public to emergencies. EAS is a digital upgrade to the old Emergency Broadcasting System. |
| **EN** | European Norms. Prefix for certain ETSI documents. |
| **environment** | A set of technologies, rules and policies which support the provision of one or more services to the end-user. |
| **EPG** | See Electronic Program Guide (EPG). |
| **ES** | See Elementary Stream (ES). |
| **ETS** | European Telecommunications Standard. Prefix for certain ETSI documents. |
| **ETSI** | European Telecommunications Standard Institute. |
| **Events** | Events are asynchronous communication between applications and the OpenCable system on which they are being executed. They provide communication between solution elements.<br><br>An event may also refer to a unit of programming, such as a movie, an episode of a television show, a newscast, or a sports game. |
| **Exclusively Reserved Keycode Event** | An input event that has been reserved by a single application and may not be reserved or received by any other application. |
| **Execution Engine (EE)** | The Execution Engine is a platform-independent interface that permits programmatic content as part of the OpenCable Application Platform |
| **FCC** | See Federal Communications Commission (FCC). |
| **Federal Communications Commission (FCC)** | A U.S. Federal agency responsible for establishing policies to govern interstate and international communications. |

| | |
|---|---|
| **Function** | A function is a process which conveys or transforms data in a predictable way. It may be affected by hardware, software, or a combination of the two. |
| **GEM** | See Globally Executable MHP |
| **Globally Executable MHP** | A terminal specification based on MHP that enables applications to interoperate across OCAP, MHP and other GEM based platforms. |
| **Global System for Mobile Communications (GSM)** | An international standard, developed in Europe, for digital mobile communications. |
| **GSM** | See Global System for Mobile Communications (GSM). |
| **HAVi** | See Home Audio/Video interoperability (HAVi) architecture. |
| **Home Audio/Video interoperability (HAVi) architecture** | A specification defined by a consumer electronics industry consortium. It is composed of a set of API's allowing for the development of applications for a home networked environment. |
| **Head-end** | The head-end refers to the control center of a cable television system, where incoming signals are amplified, converted, processed, and combined into a common cable, along with any origination cable-casting, for transmission to customers. |
| **HDTV** | See High Definition Television (HDTV). |
| **High Definition Television (HDTV)** | Television that substantially exceeds NTSC, PAL or SECAM in resolution and quality. |
| **home environment** | The environment to which an application belongs. |
| **Host device** | The host device refers to the set-top or receiver containing and executing the OpenCable Application Platform implementation. It is also host to the CableCARD device. |
| **Host device manufacturer application** | An application supplied by the host device manufacturer that is an OCAP application and a non-cable application. |
| **HTTP** | See Hypertext Transport Protocol (HTTP). |
| **Hypertext Transport Protocol (HTTP)** | HTTP is the transport layer for HTML documents over the Internet Protocol (IP). |
| **IETF** | See Internet Engineering Task Force (IETF) |
| **International Organization for Standardization (ISO)** | An international standards body. |
| **International Telecommunication Union (ITU)** | An international organization within which governments and the private sector coordinate global telecom networks and services. |

| | |
|---|---|
| **Interactive Television (ITV)** | A catch all phrase for services/platforms that allow TV viewers to interact with their television. Typical services might include interactive program guides and email and web browsing on the TV. |
| **Internet Engineering Task Force (IETF)** | A cooperative consortium that standardizes internet protocols, naming and other communications issues. |
| **Internet Protocol (IP)** | A layer 3 communications protocol commonly used in the internet. It is defined by RFC791. |
| **IP** | See Internet Protocol (IP) |
| **ISO** | See International Organization for Standardization (ISO). |
| **ITU** | See International Telecommunication Union (ITU). |
| **ITV** | See Interactive Television (ITV). |
| **Java™ API** | The Java API is a standard interface for use by platform-independent application software. It is expressed in the Java language. |
| **Java Development Kit (JDK)** | A set of resources, including software and documents, provided by SUN Microsystems to enable developers to program in the Java language. |
| **Java Media Framework (JMF)** | A Java package providing functionality primarily for data streaming. |
| **Java Secure Socket Extension (JSSE)** | A Java package providing functionality for secure network communications. |
| **JDK** | See Java Development Kit (JDK). |
| **JFIF** | See JPEG File Interchange Format ([JFIF]). |
| **JMF** | See Java Media Framework (JMF). |
| **JPEG File Interchange Format (JFIF [JFIF])** | A platform-agnostic JPEG file format. |
| **JSSE** | See Java Secure Socket Extension (JSSE). |
| **Key handling epoch** | The time interval within the application processing that starts with the delivery of a UI event to the application and ends with the notification to the platform that the application has completed all actions that may either cause the platform to forward that same key to another application, or influence how the platform will handle the subsequent key. In the case of the Execution Engine, this notification consists of returning from a key handling callback. |
| **Lifetime of an application** | The lifetime of an application characterizes the time from which the application is loaded to the time the application is destroyed. |
| **LMDS** | See Local Multipoint Distribution System (LMDS) |

| | |
|---|---|
| **Local Multipoint Distribution System (LMDS)** | A fixed wireless technology that is one solution for bringing high-bandwidth services to homes and offices within the "last mile" of connectivity. |
| **Locator** | This term has different definitions depending on the application format. For the purpose of this specification, the Locator interface provides an opaque reference to the location information of objects which are addressable within the OCAP 1.1 environment. |
| **MA** | See Monitor Application (MA). |
| **MAC** | See Media Access Control (MAC) |
| **Man-Machine Interface (MMI)** | Another term for User Interface. For the purpose of this specification, the term MMI specifies the protocol used over the CableCARD/Host interface to enable the CableCARD device to display messages on the television display. |
| **Mandatory Ordinary Keycodes** | The Mandatory Ordinary keycodes are specific key codes that can't be filtered by OCAP event filtering. |
| **Manufacturer environment** | one particular example of a non-cable environment, that is provided by the manufacturer of the CE device |
| **Media Access Control (MAC)** | A component of a networking software stack. In the OSI 7-layer model, the Media Access Control is a part of layer 2, the data link layer. |
| **Metrics** | Metrics are a set of well-defined data that are reported to a cable network by a receiver, for the purpose of measuring certain aspects of cable service delivery. |
| **MIME** | See Multipurpose Internet Mail Extensions (MIME). |
| **MHP** | See Multimedia Home Platform (MHP). |
| **MHP connected resource** | A MHP connected resource is a resource used as part of the MHP which, on its own, does not conform to the specification but which is connected to an MHP terminal in such a way that the whole is part of the MHP. |
| **MHP solution** | The MHP solution encompasses the whole set of technologies necessary to implement the MHP including protocols and APIs. |
| **MHP terminal** | An MHP terminal is a single piece of physical equipment conforming to the MHP specification, in particular in that it contains a Virtual Machine and an instance of the MHP API. |
| **MMDS** | See Multipoint Microware Distribution System (MMDS). |
| **MMI** | See Man-Machine Interface (MMI). |
| **MPEG** | A prefix for a set of file formats pertaining to video and audio data. See Moving Picture Expert Group (MPEG). |
| **Monitor Application (MA)** | The Monitor Application is a special unbound application with access to a privileged API set that manages the execution of all applications in the receiver. |

| | |
|---|---|
| **Moving Picture Expert Group (MPEG)** | The ISO working group responsible for defining the various MPEG file formats. |
| **MSO** | See Multiple Systems Operator (MSO). |
| **Multimedia Home Platform (MHP)** | The Multimedia Home Platform (MHP) consists of an MHP viewer terminal, including all possible low-to-high functionality implementations, its associated peripherals, and the in-home digital network. |
| **Multiple Systems Operator (MSO)** | A cable operator who owns more than one system. This term is often generically used for any cable operator. |
| **Multipoint Microware Distribution System (MMDS)** | A wireless broadband technology for Internet access. |
| **Multipurpose Internet Mail Extensions (MIME)** | A specification for formatting non-ASCII data for transport over the Internet. |
| **National Cable Telecommunications Association (NCTA)** | The principal trade association of the cable television industry in the US. |
| **National Television Systems Committee (NTSC)** | The NTSC are the developers of a color television system that has been adopted as a national standard. |
| **Native application** | A native application is an application written in or compiled to the machine code for the particular processor of the OCAP 1.1 device. Typically, it is written in C, C++, or assembly language and may be supplied with the OCAP 1.1 implementation or downloaded over the cable. Native applications are normally non-cable applications. |
| **Native library** | A native library is a library written in or compiled to the machine code for a particular processor. Typically, it is written in C, C++, or assembly language. |
| **Navigator** | A navigator is a resident application, typically provided by the manufacturer, that the end user can activate at any time. The navigator can be used to select services, applications, and initiate interoperable applications. |
| **NCTA** | See National Cable Telecommunications Association (NCTA). |
| **Near-Video-On-Demand (NVOD)** | Video-On-Demand with which the user may experience some delay before content begins. |
| **Non-cable application** | Application deployed to a OCAP host device by means outside the scope of OCAP, e.g., supplied with the OCAP implementation by the manufacturer of the OCAP host device or other party. |
| **non-cable environment** | A set of technologies rules and policies other than that provided by the OCAP specification and the monitor application. |
| **Non-OCAP application** | Application which cannot be controlled by the OCAP monitor application. |

| | |
|---|---|
| **Non-Reserved Keycode Event** | An input event that is to be delivered to the application with input focus. |
| **non-shared resource** | Resources which are either always controlled by the cable environment (e.g., the integrated cable return channel) or which are never controlled by the cable environment (e.g., a terrestrial tuner). For a list of these resources see Section 19.2.1.2. |
| **NTSC** | See National Television Systems Committee (NTSC). |
| **NVOD** | See Near-Video-On-Demand (NVOD). |
| **Object carousel** | An object carousel is a repetitively broadcast file system. |
| **OCAP** | See OpenCable Application Platform (OCAP). |
| **OCAP application** | Application which can be controlled by the OCAP monitor application. Among other things, OCAP applications are listed in the applications database and observe the OCAP resource management mechanisms for both shared and OCAP resources. |
| **OCAP-J** | OCAP-J is an application type. |
| **OCAP-J proxy application** | An OCAP-J application that is associated with a native application and which serves as a proxy for the native application in the OCAP domain for purposes of lifecycle management, resource management, and access control. |
| **OCAP 1.1 implementation** | The OCAP 1.1 implementation is the actual software that provides support for the defined OCAP 1.1 on a host receiver. |
| **OCAP 1.1 API** | OCAP API refers to one of the Java APIs standardized as part of the OCAP 1.1 Profile. For the OCAP 1.1 implementation, the OCAP 1.1 APIs include the DVB-J APIs that have been modified and/or extended by this specification. |
| **OOB channel** | An OOB, or out-of-band, channel is the combination of the forward and reverse out-of-band communications channels. The OOB channel provides an IP-based communication channel between the network and the digital set-top converter. |
| **OOB-FDC** | See Out-Of-Band-Forward-Data-Channel (OOB-FDC). |
| **OOB-RDC** | See Out-Of-Band-Reverse-Data-Channel (OOB-RDC). |
| **OpenCable Application Platform (OCAP)** | The OpenCable Application Platform is a software interface specification that completely defines the OpenCable host software interface that executes OpenCable portable applications. |
| **OpenCable device** | An OpenCable-compliant digital set-top converter or cable ready digital television, allowing reception of existing cable television channels and providing the user interface for future, interactive applications. |
| **OpenCable terminal** | See OpenCable device. |
| **OS** | See Operating System (OS). |

| | |
|---|---|
| **Operating System (OS)** | The software that controls the underlying hardware, performs the most basic functions for managing the resources of the hardware, and provides services to other software such as applications. |
| **Out-Of-Band-Forward-Data-Channel (OOB-FDC)** | The portion of the cable RF range that is used to deliver system or service information to a receiver. Its frequency range is generally 70-130Mz. |
| **Out-Of-Band-Reverse-Data-Channel (OOB-RDC)** | The portion of the cable RF range that is used to deliver data from the home receiver to the head-end. Its frequency range is 5-40Mz. |
| **Packet Identifier (PID)** | MPEG-2 assigns a PID to each data packet. Packets with the same PID belong to the same logical channel. |
| **Packetized Elementary Streams (PES)** | An MPEG stream is composed of one or more elementary streams (ES), each containing audio, video, or data. ESs can be grouped into Program Streams, which are formed by breaking ESs into chunks, the PESs, and interleaving them. |
| **Pause-able application** | An OCAP application that has an implementation of its pauseXlet and startXlet methods such that it can handle being repeatedly paused and restarted. |
| **Pay-Per-View (PPV)** | A service that allows customers to buy content on a program by program basis. Most customers today order programming via phone. OCAP 1.1 systems will enable the development of services to allow customers to order pay-per-view programming directly from their TV. |
| **PCR** | Program Clock Reference. |
| **Personal Identification Number (PIN)** | Unique number used for security purposes to verify the identity of an individual user on a specific network. |
| **Personal Video Recorder (PVR)** | A set of equipment that allows a user to timeshift television without removable media. |
| **PES** | See Packetized Elementary Steams (PES). |
| **PID** | See Packet Identifier (PID). |
| **PIN** | See Personal Identification Number (PIN). |
| **Platform Reserved Keycode Event** | An input event that has been reserved exclusively and permanently by a manufacturers' application. |
| **Plug-in** | A plug-in refers to a set of functionality which can be added to a generic platform in order to provide interpretation of DVB registered, but non-DVB-J, application formats. For example, HTML3.2 or MHEG-5 are examples of plug-ins. |
| **Plug-in application** | A plug-in application refers to an application that conforms to an application format for which a plug-in has been registered with DVB and which is only interoperable within terminals which have the appropriate plug-in resident or connected to networks where an appropriate plug-in is being broadcast. |
| **PMT** | See Program Map Table (PMT). |

| | |
|---|---|
| **POD** | See Point Of Deployment (POD). |
| **Point Of Deployment (POD)** | A POD, also referred to as a CableCARD device, is a detachable device distributed by cable providers, that connects to the home receiver. The interface between the POD unit and the receiver is specified by the OpenCable platform. POD functionality includes copy protection and signal demodulation. |
| **PPV** | See Pay-Per-View (PPV). |
| **Profile** | A profile is a description of a series of minimum configurations, defined as part of the specification, providing different capabilities of the OpenCable system. A profile maps a set of functions which characterize the scope of service options. The number of profiles is small. The mapping of functions into resources and subsequently into hardware entities is out of the scope of the specification and is left to manufacturers. |
| **Program and System Information Protocol (PSIP)** | A transport and data format specification formulated by ATSC A65A to deliver System Information to the receiver. |
| **Program Map Table (PMT)** | This is a MPEG-2 entity that contains all of the PIDs that make up a program. |
| **PSIP** | See Program and System Information Protocol (PSIP). |
| **PSTN** | See Public Switched Telephone Network (PSTN). |
| **Public Switched Telephone Network (PSTN)** | The international telephone system based on copper wires carrying analog voice data. |
| **PVR** | See Personal Video Recorder (PVR). |
| **RCMM** | See Section 14.2.2.7, Root Certificate Management |
| **Remote Method Invocation (RMI)** | A Java programming feature that allows a program running on one computer to access the objects and methods of another Java program running on a separate computer. |
| **Request For Comment (RFC)** | A document and a process for establishing standards. IETF standards are documented as RFCs. |
| **Resident application** | A resident application is an application that is saved in the host device and may be run at any time. Resident applications, such as the Monitor Application or EPG, are typically responsible for host device control. |
| **Return channel** | A return channel refers to the communications mechanism that provides connection between OpenCable and a remote server. |
| **RFC** | See Request For Comment (RFC). |
| **RMI** | See Remote Method Invocation (RMI). |

| | |
|---|---|
| **Sandbox** | Unsigned applications and signed applications without a permission file have access to all the APIs for which there is no permission signaling defined. This is commonly called the sandbox. |
| **Satellite Master Antenna TV (SMATV)** | RF distribution of satellite and antenna signals. |
| **SCTE** | See Society of Cable Telecommunications Engineers (SCTE). |
| **Secure Sockets Layer (SSL)** | A public key encryption based protocol for secure communications between client and server. |
| **selected environment** | The single environment whose rules and policies determine the use of shared resources in a device and the operation of applications using those shared resources.<br><br>NOTE: See Section 10.2.2.4.1 for details on when and how environments become selected. |
| **Service** | A service is a sequence of programs under the control of a broadcaster which can be broadcast as part of a schedule. |
| **Service application** | An application is service-bound if, and only if, it is associated with one or more broadcast services. |
| **Service Information (SI)** | That information that describes the broadcast services available on the network. |
| **Shared Reserved Keycode Event** | An input event that has been reserved by more than one application. |
| **shared resource** | Resources which can be used both in the cable environment and in non-cable environments. For a list of these resources see Section 19.2.1.2. |
| **SI** | See Service Information (SI). |
| **SMATV** | See Satellite Master Antenna TV (SMATV). |
| **Society of Cable Telecommunications Engineers (SCTE)** | A non-profit organization, dedicated to advancing the careers of broadband's engineering and operations professionals, by promoting member services and resources, focused on three key areas: Professional Development, Information, and Standards. |
| **SSL** | See Secure Sockets Layer (SSL). |
| **TLS** | See Transport Layer Security (TLS). |
| **Transport Layer Security (TLS)** | An Internet security protocol based on SSL. |
| **Transient application** | A transient application is an application that SHALL be downloaded before it can be run and may be deleted afterward. Transient applications, such as a program enhancement, are typically delivered via the broadcast stream or by request if 2-way functionality is present. |

| | |
|---|---|
| **Trigger** | A trigger is an event that may cause a change in the behavior of an application that registers interest in such events. Triggers may come from many sources (for example, 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. |
| **UI** | See User Interface (UI). |
| **Unbound application** | An unbound application is not associated with a broadcast service. |
| **User Interface (UI)** | Also known as UI, a user interface is the sensory and behavioral aspects of a program that are presented to a user. The term is generally used to denote the menuing and navigational constructs of a program. |
| **UTC** | See Coordinated Universal Time (UTC). |
| **Utility applications** | Cross-environment applications that are responsible for setup and control of fundamental properties and characteristics of the iDCR, e.g., volume control. |
| **VBI** | See Vertical Blanking Interval (VBI). |
| **Vertical Blanking Interval (VBI)** | A portion of the television signal that does not contain visual data. In NTSC, the VBI are lines 1 through 21 in each field. |
| **VCT** | See Virtual Channel Table (VCT). |
| **Video-On-Demand (VOD)** | A television service where viewers can select and watch video content for viewing at any time. |
| **Virtual Channel Table (VCT)** | Data declared as part of the Service Information standard defined by SCTE. |
| **VOD** | See Video-On-Demand (VOD). |
| **XAIT** | Extended Application Information Table. Used for launching and managing the lifecycle of unbound applications. |
| **Xlet** | Xlet is the interface used for Execution Engine application lifecycle control. |

# 4 ACRONYMS

This section contains requirements that are extensions to [DVB-GEM 1.1] Section: 3 Definitions and abbreviations and are extensions of [DVB-MHP1.1.2] Section: 3 Definitions and abbreviations.

## 4.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section. 4 (this section) corresponds to [DVB-MHP1.1.2] and [DVB-GEM 1.1] as follows:

***Table 4–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 4 Acronyms | 3 Definitions and abbreviations | Extension | 3 Definitions and abbreviations | Extension |
| 4.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 4.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 4.2.1 Acronyms | 3.2 Abbreviations | Extension | 3.2 Abbreviations | Extension |

## 4.2 OCAP 1.1 Specific Requirements

This subsection is an OCAP specific section that does not correspond to any [DVB-GEM 1.1] Section.

### 4.2.1 Acronyms

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 3.2 Abbreviations and are extensions of [DVB-MHP1.1.2] Section: 3.2 Abbreviations.

The following acronyms are used in this specification:

**AIT** Application Information Table

**API** Application Program Interface

**ATSC** Advanced Television System Committee

**AWT** Abstract Windowing Toolkit

**BFS** Broadcast File System.

**BNF** Backus-Naur Form

**CA** Conditional Access

**CATV** Cable TV

| | |
|---|---|
| **CCI** | Copy Control Information. |
| **CHILA** | CableCARD Host Interface License Agreement |
| **CORBA** | Common Object Request Broker Architecture |
| **CP** | Content Protection |
| **CRC** | Cyclic Redundancy Check |
| **CRL** | Certificate Revocation List |
| **DAVIC** | Digital Audio Visual Council |
| **DECT** | Digital Enhanced Cordless Telecommunications |
| **DHCP** | Dynamic Host Configuration Protocol |
| **DII** | DownloadInfoIndication |
| **DOCSIS** | Data-Over-Cable Service Interface Specification |
| **DPI** | Digital Program Insertion |
| **DSM-CC** | Digital Storage Media—Command and Control. Part 6 of the MPEG-2 standard |
| **DSM-CC-OC** | Digital Storage Media—Command and Control Object Carousel |
| **DSM-CC-UU** | Digital Storage Media—Command and Control User to User |
| **DTD** | Document Type Definition |
| **DVB** | Digital Video Broadcasting |
| **DVS** | Digital Video Subcommittee |
| **DTVCC** | Digital Television Closed Captioning |
| **EAS** | Emergency Alert System |
| **ECM** | Entitlement Control Message |
| **EE** | Execution Engine |
| **EIA** | Electronic Industry Alliance |
| **EN** | European Norms |
| **EPG** | Electronic Program Guide |
| **ES** | Elementary Stream |
| **ETS** | European Telecommunications Standard |

| | |
|---|---|
| **ETSI** | European Telecommunications Standard Institute |
| **ExCCI** | Extended Copy Control Information |
| **FCC** | Federal Communications Commission |
| **GEM** | Globally Executable MHP |
| **GSM** | Global System for Mobile Communications |
| **HAVi** | Home Audio/Video interoperability. |
| **HDNI** | Home Digital Network Interface |
| **HDTV** | High Definition Television |
| **HTML** | Hypertext Mark-up Language |
| **HTTP** | Hypertext Transport Protocol |
| **IETF** | Internet Engineering Task Force |
| **IP** | Internet Protocol |
| **IPDR/SP** | Internet Protocol Detail Record/Streaming Protocol. The transmission protocol used to transmit metrics to a cable operator. |
| **ISO** | International Organization for Standardization |
| **ITU** | International Telecommunication Union |
| **ITV** | Interactive Television |
| **JDK** | Java Development Kit |
| **JFIF** | JPEG File Interchange Format |
| **JMF** | Java Media Framework |
| **JSSE** | Java Secure Socket Extension |
| **LMDS** | Local Multipoint Distribution System |
| **MA** | Monitor Application |
| **MAC** | Media Access Control |
| **MHP** | Multimedia Home Platform |
| **MMDS** | Multipoint Microware Distribution System |
| **MMI** | Man Machine Interface |

| | |
|---|---|
| **MPEG** | Moving Picture Expert Group |
| **MSO** | Multiple Service Operator |
| **MIME** | multipurpose internet mail extensions |
| **NCTA** | National Cable Telecommunications Association |
| **NPT** | Normal Play Time |
| **NTSC** | National Television Systems Committee |
| **NVOD** | Near-Video-On-Demand |
| **OCAP** | OpenCable Application Platform |
| **OOB channel** | Out-Of-Band channel |
| **OOB-FDC** | Out-Of-Band Forward-Data-Channel |
| **OOB-RDC** | Out-Of-Band Reverse-Data-Channel |
| **OS** | Operating System |
| **PCR** | Program Clock Reference |
| **PES** | Packetized Elementary Stream |
| **PID** | Packet Identification number |
| **PMT** | Program Map Table |
| **POD** | Point of Deployment module |
| **PPV** | Pay-per-view |
| **PSIP** | Program and System Information Protocol |
| **PSTN** | Public Switched Telephone Network |
| **PVR** | Personal Video Recorder |
| **RCMM** | Root Certificate Management Messages |
| **RMI** | Remote Method Invocation |
| **SCTE** | Society of Cable Telecommunications Engineers |
| **SDP** | Session Description Protocol |
| **SI** | Service Information |
| **SMATV** | Satellite Master Antenna TV |

| | |
|---|---|
| **SSL** | Secure Sockets Layer |
| **STC** | System Time Clock |
| **TLS** | Transport Layer Security |
| **UI** | User Interface |
| **URI** | Universal Resource Identifier |
| **URL** | Uniform Resource Locator |
| **UTC** | Coordinated Universal Time |
| **VBI** | Vertical Blanking Interval |
| **VCT** | Virtual Channel Table |
| **VOD** | Video-On-Demand |
| **XAIT** | Extended Application Information Table |

# 5 CONVENTIONS

This section contains requirements that are extensions to [DVB-GEM 1.1] Section: 4 General considerations and conventions.

The following conventions are used in this manual:

## 5.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that does not correspond to any [DVB-GEM 1.1] Section.

***Table 5–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| 5 Conventions | 4 General considerations and conventions | Extension |
| 5.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension |
| 5.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension |
| 5.2.1 Specification Language | No Corresponding Section | OCAP-Specific Extension |
| 5.2.2 Organization of the OCAP Specification | No Corresponding Section | OCAP-Specific Extension |
| No Corresponding Section | 4.1 General considerations | Compliance |
| No Corresponding Section | 4.2 Conventions | Compliance |

## 5.2 OCAP 1.1 Specific Requirements

This subsection is an OCAP specific section that does not correspond to any [DVB-GEM 1.1] Section.

### 5.2.1 Specification Language

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

The following words are used throughout this document to define the significance of particular requirements:

| | |
|---|---|
| SHALL | This word, or the adjective REQUIRED, means that the item is an absolute requirement of this specification. |
| SHALL NOT | This phrase means that the item is an absolute prohibition of this specification. |
| SHOULD | This word, or the adjective RECOMMENDED, means that valid reasons may exist in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course. |
| SHOULD NOT | This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label. |

| | |
|---|---|
| MAY | This word, or the adjective OPTIONAL, means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item. |

**Note:** Other text is descriptive or explanatory.

### 5.2.2 Organization of the OCAP Specification

This document uses the [DVB-GEM 1.1] Specification as its base. Where applicable, OCAP 1.1 references the corresponding section within [DVB-GEM 1.1]. However, this document does not follow the same organization as the [DVB-GEM 1.1] document. Consequentially, there is not a one-to-one correspondence to sections and annexes.

The correspondence of OCAP sections to DVB-GEM sections will be defined in the OCAP section, and in conformance Tables included in each Major Section of the document. Generally, there will be one of the following conformance statements in each section:

- Any conformance statement in a section will apply to its subsections as well unless that subsection has its own conformance statement.
- Sections of DVB-GEM which are not referred to in this document are assumed to be conformed to by OCAP, and should be so indicated in the Conformance Tables (defined below).
- DVB-GEM Specifications typically refer to DVB-MHP Specifications, by default the OCAP document will refer to the DVB-GEM specification and not make any mention to the DVB-MHP section. Conformance to such DVB-MHP sections is inferred accordingly through the OCAP Compliance to the DVB-GEM specification, and the DVB-GEM use of the DVB-MHP Section.
- The OCAP document will refer directly to DVB-MHP sections in the following cases:
- All API packages are defined in annexes. Most non-API information is provided in the main specification. The exceptions to this are errata and informative references.
- Information in all of these sections is assumed to be Normative, unless explicitly indicated as "Informative".

#### *5.2.2.1 Required Sections and Subsections in the OCAP Document.*

Each of the major sections (chapters) of the OCAP Specification will have the following sequence of sections:

<<Major Section Name>>

x.1 DVB-GEM and DVB-MHP Correspondence

x.2 DVB-GEM and DVB-MHP Extensions and Deviations

x.3 OCAP Specific Requirements

Each of these sections are defined below.

##### *5.2.2.1.1 Form of the "DVB-GEM and DVB-MHP Correspondence" Section*

The correspondence Tables will follow the following conventions:

1. All GEM Sections should be represented in the Tables.

    a. If OCAP is completely in compliance with a GEM section, GEM subsections do not need to be in the Table - they are assumed to be compliant as well.

b. If OCAP modifies any subsection within a GEM Section, every subsection of that section SHALL be represented in the Table.

2. All OCAP Sections should be represented in the Tables.

   a. When a section with subsections either is completely compliant with GEM and MHP or has no corresponding section in GEM or MHP only that section needs to be in the Table.

   b. If any subsection of a section modifies or extends GEM (or MHP) it SHALL be in the Table, and all other subsections of the same parent section SHALL be in the Table.

3. DVB-MHP 1.1.2 sections will only be included in a Table when they are directly changed by OCAP.

   a. Normally OCAP will refer directly to a DVB-GEM specification, and conformance will be assumed to be the same according to the DVB-GEM Specification.

   b. In certain cases (such as when DVB-MHP presents an Implementation of DVB-GEM requirements) OCAP will deviate from or extend DVB-MHP. In that case, the conformance Tables will include an MHP Column that will be expanded according to the same rules of the GEM sections.

The format of the Table will be:

| OCAP 1.1 Section | [DVB-GEM 1.1] Section | Compliance to DVB-GEM | [DVB-MHP1.1.2] Section | Compliance to DVB-MHP |
|---|---|---|---|---|
| <<section number and name>> or "No Corresponding Section" | <<section number and name>> or "No Corresponding Section>> | "Complete Compliance", or "Extension", or "Deviation" | <<section number and name - if referred to by OCAP>> or empty | "Complete Compliance", or "Extension", or "Deviation" |

The definition of the compliance terms is as follows:

   a. Compliance - is used when OCAP exactly complies with specifications cited.

   b. Extension - is used when OCAP provides additional specifications to the specification cited.

   c. OCAP Specific Requirement - is used when there is no direct GEM or MHP specification indicated in a requirement.

# 6 GEM AND MHP CORRESPONDENCE

This section contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

## 6.1 GEM Compliance

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

OCAP terminals SHALL comply in full with [DVB-GEM 1.1]. For avoidance of doubt, in the event of a conflict between [DVB-GEM 1.1] and this specification, the normative guarantees of [DVB-GEM 1.1] shall take precedence except as detailed in Section 6.1.1.

Because OCAP 1.1 is a GEM Terminal Specification (as defined in [DVB-GEM 1.1]), the following terminology applies:

- An OCAP 1.1 terminal is a kind of GEM terminal
- A GEM application is a kind of OCAP application

**Note:** (informative) OCAP applications that use OCAP extensions to GEM are not GEM applications.

**Note:** (informative) As specified in Section 17, compliance with the interactive broadcast profile of [DVB-GEM 1.1] is required.

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| 6 GEM and MHP Correspondence | No Corresponding Section | OCAP-Specific Extension |
| 6.1 GEM Compliance | No Corresponding Section | OCAP-Specific Extension |
| 6.2 DVB-GEM and DVB-MHP Full Compliance (informative) | No Corresponding Section | OCAP-Specific Extension |

### 6.1.1 GEM Errata

Following are errata to [DVB-GEM 1.1]. The changes presented have been agreed by the appropriate DVB subgroup for publication in a future version of GEM.

#### *6.1.1.1 VK_TELETEXT input event optional*

[DVB-GEM 1.1] clause G is considered to read:

> MHP [ATSC A/52B], clause G.5 is included in the present document with the following modification:
>
> GEM-based terminal specifications are allowed to make optional the VK_TELETEXT input event.

#### *6.1.1.2 SelectPermission*

The following additional paragraphs are considered to be present at the end of GEM clause 11.10 "Java permissions":

> GEM terminal specifications may define an alternative to MHP clause 11.10.2.7, "javax.tv.service.selection.SelectPermission" in order to narrow the scope to apply only to broadcast services.

Hence, applications will have the permissions needed to select all broadcast services in their own service context(s) unless denied by an entry in the permission request file.

**Note:** GEM terminal specifications may define types of service other than broadcast. The present document allows replacement of MHP clause 11.10.2.7 to only apply to broadcast services."

## 6.2 DVB-GEM and DVB-MHP Full Compliance (informative)

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

As a consequence of Section 6.1, OCAP 1.1 is fully compliant with the following [DVB-GEM 1.1] and [DVB-MHP1.1.2] sections. This section maps out the sections of the OCAP 1.1 profile that are not specifically called out in the compliance Tables in the individual sections of the OCAP 1.1 profile to the [DVB-GEM 1.1] and [DVB-MHP1.1.2] specifications.

Unless called out in the following Table, a compliant section infers that all subsections are also compliant. For example, if 5.2 is listed as compliant, then 5.2.1 and 5.2.2 are also compliant, but not listed:

***Table 6–1 - Compliance Table***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] | MHP Compliance |
|---|---|---|---|---|
| Annex A (deleted) | Annex A (normative): External references; errata, clarifications and exemptions | Compliance | Annex A (normative): External references; errata, clarifications and exemptions | Compliance |
| No Corresponding Section | Annex E (normative): Character set | Compliance | Annex E (normative): Character set | Compliance |
| No Corresponding Section | Annex F (informative): Authoring and implementation guidelines | Compliance | Annex F (informative): Authoring and Implementation Guidelines | Compliance |
| No Corresponding Section | Annex K (normative): DVB-J persistent storage API | Compliance | Annex K (normative): DVB-J persistent storage API | Compliance |
| No Corresponding Section | Annex L (normative): User settings and preferences API | Compliance | Annex L (normative): User Settings and Preferences API | Compliance |
| No Corresponding Section | Annex M (normative): SI Access API<br>Note: GEM Annex M does not include MHP Annex M and specifies that a functional equivalent be defined instead. | Compliance | | |
| No Corresponding Section but see Section 13.3.7 | Annex P (normative): Broadcast transport protocol access | Compliance | Annex P (normative): Broadcast Transport Protocol Access | Compliance |
| No Corresponding Section | Annex U (normative): Extended graphics APIs | Compliance | Annex U (normative): Extended graphics APIs | Compliance |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] | MHP Compliance |
|---|---|---|---|---|
| No Corresponding Section | Annex W (informative): DVB-J examples | Compliance | Annex W (normative): DVB-J examples | Compliance |
| No Corresponding Section | Annex X (normative): Test support | Compliance | Annex X (normative): Test support | Compliance |
| No Corresponding Section | Annex Y (normative): Inter-application and Inter-Xlet communication API | Compliance | Annex Y (normative): Inter-application communication API | Compliance |
| No Corresponding Section | Annex Z (informative): Services, service contexts and applications in an MHP environment | Compliance | Annex Z (informative): Services, Service Contexts and Applications in an MHP Environment | Compliance |
| No Corresponding Section | Annex AF (normative) Plug-in APIs | Compliance | Annex AF (normative) | Compliance |
| No Corresponding Section | Annex AG (normative) Stored application APIs | Compliance | Annex AG (normative) Stored application APIs | Compliance |
| No Corresponding Section | Annex AH (normative) Internet client APIs | Compliance | Annex AH (normative) Internet client APIs | Compliance |
| No Corresponding Section | Annex AK (normative) Extended service Selection API | Compliance | Annex AK (normative) Extended service Selection API | Compliance |
| No Corresponding Section | Annex AL (normative) Extended content referencing API | Compliance | Annex AL (normative) Extended content referencing API | Compliance |

When particular sections of the GEM or DVB-MHP Specifications that are referenced here are read in the context of this specification, the phrase OCAP 1.1 application should be substituted for the phrase GEM application or MHP application. Similarly, OCAP 1.1 environment should be substituted for GEM environment or MHP environment and the term OCAP 1.1 should be read as a substitute for the term GEM or MHP when such term is used to refer to the platform, itself.

When reading these sections, OCAP 1.1 replaces GEM and MHP; for example, in the following terms and phrases:

- GEM terminals should be replaced with OCAP 1.1 terminals.
- Applications should be replaced with OCAP 1.1 applications.
- Broadcast GEM applications should be replaced with the phrase broadcast service applications.

These sections in [DVB-GEM 1.1] and [DVB-MHP1.1.2] are not reprinted in the OCAP 1.1 Profile. Corresponding OCAP 1.1 Profile sections are not written.

# 7 BASIC ARCHITECTURE

This section is compliant with [DVB-GEM 1.1] Section: 5 Basic architecture and are extensions of [DVB-MHP1.1.2] Section: 5 Basic Architecture.

The OCAP 1.1 Profile identifies a software Application Programming Interface (API) for use on OpenCable retail host devices. This API serves as an application platform that is independent of the underlying hardware. Likewise, the OCAP 1.1 API provides an abstraction layer to the set-top box's operating system that manages the underlying hardware's resources.

This section of the OCAP 1.1 Profile describes how the OCAP 1.1 software stack is organized.

## 7.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section. Section 7 (this section) of the OCAP 1.1 Profile corresponds to Section 5 of [DVB-MHP1.1.2] as follows ([DVB-GEM 1.1] does not define an architecture):

***Table 7–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 7 Basic Architecture | 5 Basic Architecture | Compliance | 5 Basic Architecture | Extension |
| 7.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 7.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 7.2.1 Deviations from DVB-MHP | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 7.2.1.1 Context | 5 Basic architecture | Compliance | 5.1 Context. | Extension |
| 7.2.1.2 Architecture | 5 Basic architecture | Compliance | 5.2 Architecture | Extension |
| No Corresponding Section | 5 Basic architecture | Compliance | 5.2.1 Resources | Compliance |
| No Corresponding Section | 5 Basic architecture | Compliance | 5.2.2 System software | Compliance |
| 7.2.1.3 Application | 5 Basic architecture | Compliance | 5.2.3 Application | Extension |
| No Corresponding Section | 5 Basic architecture | Compliance | 5.3 Interfaces Between an MHP Application and the MHP System | Compliance |
| 7.2.1.4 Plug-ins | 5 Basic architecture | Compliance | 5.4 Plug-ins | Extension |
| 7.2.1.5 Informative | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |

## 7.2 OCAP 1.1 Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

### 7.2.1 Deviations from DVB-MHP

This subsection contains OCAP-specific requirements that do not correspond to any 2 [DVB-GEM 1.1] Section.

#### *7.2.1.1 Context*

This subsection is compliant with [DVB-GEM 1.1] Section: 5 Basic architecture and is an extension of [DVB-MHP1.1.2] Section: 5.1 Context.

Figure 7–1 shows the context in which the OCAP 1.1 software stack will execute. In that Figure, the OCAP 1.1 software stack resides on the OCAP 1.1 Host Device.

Like MHP, the OCAP 1.1 software has access to streams of video, audio, data, and other media assets. Because OCAP 1.1 is a cable industry specification, these streams are distributed over a cable operator's network as represented by the Cable Distribution Network element in Figure 7–1.

OCAP 1.1 applications process the streams and presents content to the viewer. The viewer may interact with the application through input and output peripherals attached or associated with the OCAP 1.1 host device. The OCAP 1.1 platform will receive input from the viewer via input devices such as a remote control or keyboard. In response to viewer input, the platform will present visual output to a screen or television as well as audio output to loudspeakers.

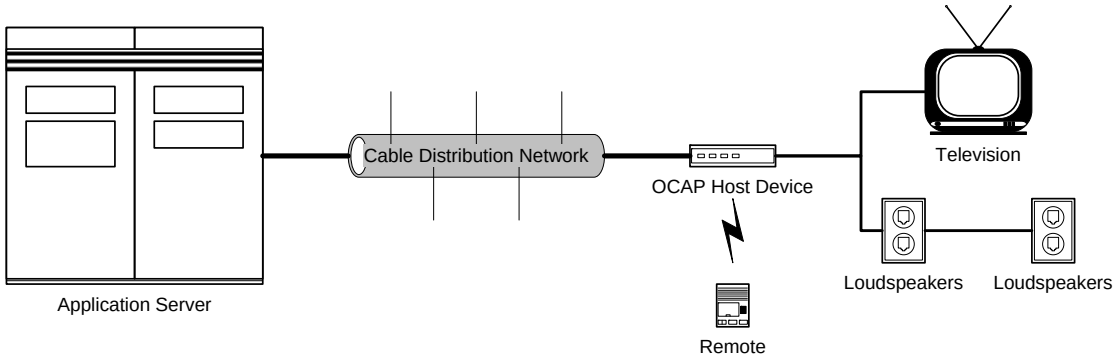



***Figure 7–1 - OCAP 1.1 Context***

The OCAP 1.1 context does not comply with the local cluster model presented in this section of the DVB-MHP specification and, thus, is not consistent with the DVB IHDN (In-Home Digital Networks) specification. A corresponding model for OCAP 1.1 is not made available in this version of the specification.

#### *7.2.1.2 Architecture*

This subsection is compliant with [DVB-GEM 1.1] Section: 5 Basic architecture and is an extension of [DVB-MHP1.1.2] Section: 5.2 Architecture.

This section describes how elements of the OCAP 1.1 architecture are organized.

The OCAP 1.1 model is similar to the DVB-MHP model in that it distinguishes between hardware entities or resources, system software, and applications. However, a more accurate description of the OCAP 1.1 architecture is described in Section 7.2.1.5.1.

#### 7.2.1.3 Application

This subsection is compliant with [DVB-GEM 1.1] Section: 5 Basic architecture and is an extension of [DVB-MHP1.1.2] Section: 5.2.3 Application

Figure 7–2 is a more accurate illustration of the OCAP 1.1 architecture than the one presented in Figure 4, Section 5.2.3 of the [DVB-MHP1.1.2].

In OCAP 1.1, the application manager is still considered to be part of the System Software layer. However the application manager SHALL be implemented to interface with the Monitor Application as defined in Section 21 of this specification.

#### 7.2.1.4 Plug-ins

This subsection is compliant with [DVB-GEM 1.1] Section: 5 Basic architecture and is an extension of [DVB-MHP1.1.2] Section: 5.4 Plug-ins.

Interoperable plug-ins are supported by OCAP 1.1. An interoperable plug-in SHALL be an OCAP-J application.

Implementation specific plug-ins are not supported by OCAP 1.1.

#### 7.2.1.5 Informative

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

The following information is informative to the OCAP 1.1 Profile. It attempts to clarify issues in the [DVB-MHP1.1.2] as well as provide additional information to the normative requirements.

##### 7.2.1.5.1 OCAP 1.1 Software Architecture

The OCAP architecture is described below to aid in understanding the APIs and software modules presented in this specification. The architecture is designed to allow cable operators a platform by which to deploy a wide variety of interactive television services on their networks. The range of services extends from simple broadcast enhancements to complex interactive environments.

The OCAP 1.1 software architecture is illustrated in Figure 7–2.

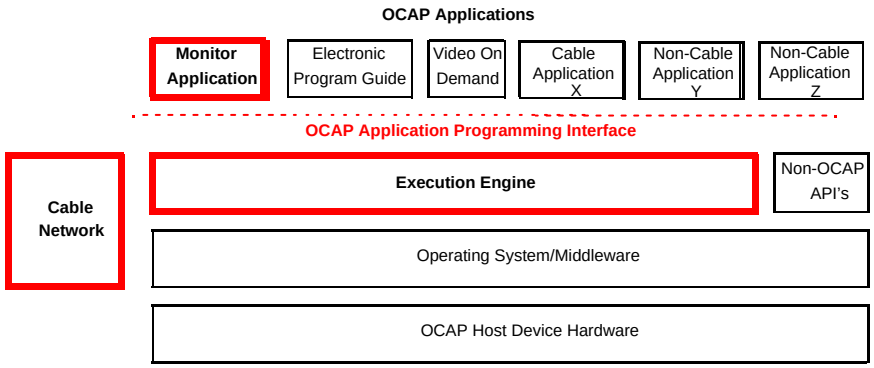



***Figure 7–2 - OCAP1.1 Software Architecture***

The architecture is divided into sections based on the responsibility of each subsystem or application and its relationship to this specification:

- Applications are written for cable operators, content providers, CE manufacturers or some combination of the three. They may implement a service or capability provided by the cable operator to the viewer.
- The subsystems highlighted in red/bold in Figure 7–2 are the OCAP 1.1 subsystems specified by this document.
- The OCAP 1.1 implementation consists of all of the subsystems shown in Figure 7–2, except for the applications including the monitor and native applications.
- The operating system/middleware subsystem manages the resources of the OCAP 1.1 host device.
- The host device subsystem represents the hardware present on a digital set-top box.

The subsystems are described in the following sections.

#### 7.2.1.5.1.1 OCAP 1.1 Host Device subsystem

OCAP 1.1 is designed for advanced digital set-top converters. These devices are actually a hybrid analog/digital set-top box. The set-top box SHALL be able to support all existing analog services, as well as the new digital services which this specification enables. Refer to Section 25.

##### 7.2.1.5.1.1.1 Host Device Major Subsystems

OCAP 1.1 SHALL be portable to a variety of hardware engines. Thus, there are a number of different configurations that will be compliant with this specification. However, the OCAP 1.1 host device will typically be comprised of the following subsystems:

- The cable network interface (CNI)
  This is the interface between the cable system and the set-top box. The CNI is based upon [SCTE 40]. The interface in the Host device consists of the cable tuner, NTSC demodulator, QAM demodulator, and an out-of-band (OOB) receiver/transmitter.
- Conditional Access (CA)
  CA is the means by which access to specific services is granted to the customer. It performs the equivalent of analog de-scrambling for digital services.
- Video and Graphics Processing
  This subsystem is responsible for MPEG-2 decoding and on-screen display generation. OCAP 1.1 specifies a multi-plane architecture consisting of a Video plane and a Graphics plane.
- Audio Processing
  This subsystem is responsible for decoding AC3 and other audio formats.
- Central Processing Unit (CPU) and RAM
  The CPU subsystem is the central processing unit of the set-top box. It includes a microprocessor together with ROM, Flash, and RAM memory.
- Inputs
  The viewer communicates with the OCAP 1.1 host device via input devices such as an IR remote control or keyboard.
- Outputs
  In response to viewer input, the OCAP 1.1 host device presents output via base-band (or composite) video and base-band audio. Optional outputs may include component video (e.g., YPrPb), S-Video, digital audio (S/PDIF), and the IEEE-1394 digital interface.

#### 7.2.1.5.1.2 Operating System Middleware

The operating system middleware subsystem is native software written to support the set-top hardware and to provide basic services.

The OpenCable Application Platform SHALL be portable to a variety of operating systems. Thus, there are a number a different configurations that will be compliant with this specification. However, the operating system middleware will typically be comprised of the following subsystems:

- Task/Process Scheduling
  This subsystem ensures that the applications each have a turn to use the CPU and are not "starved" or blocked from executing.
- Interrupt Handling
  This module dispatches interrupts to the appropriate devices for managing real-time events.
- Device Drivers
  These are the software modules designed to initialize and manage the devices available on the set-top box.
- Memory Management
  This module manages the OCAP 1.1 host device's memory resources.
- Timers
  Timers are used to trigger task execution to support task/process scheduling and dispatching.
- Synchronization
  This module provides a means by which to access and synchronize shared resources.

#### 7.2.1.5.1.3 OCAP 1.1 Subsystems

The OCAP 1.1 subsystems highlighted in red/bold in Figure 7–2 are the modules addressed by this specification. These modules can be classified as Cable Network, the Execution Engine, and the Monitor Application.

##### 7.2.1.5.1.3.1 Cable Network

The cable network sub-system includes protocols and behavior to support cable networks. Network protocols include:

1. application protocols for communicating between application components that are distributed from the Host device to other network locations,
2. cable network protocols for audio/video, data including applications, and system information,
3. Host support for CableCARD interface/Host resources,

Examples of specified behaviors include, but are not limited to CableCARD initialization, response to system information, closed captioning, and emergency alert system.

##### 7.2.1.5.1.3.2 Execution Engine (EE)

The Execution Engine (EE) is an implementation of the software that delivers part of the OCAP 1.1. It provides a platform-independent interface that permits programmatic content from various service and application providers to run on the hardware and software implementations of various manufacturers. The Execution Engine is based upon Java technologies from Sun Microsystems and includes the Java Virtual Machine and a collection of Java APIs that abstract the functionality of the specific hardware and software. Applications intended for the Execution Engine are compiled into Java byte code format and executed by the Java Virtual Machine. An application running in the Execution Engine uses the provided Java APIs to access the television functionalities provided by the receiver.

A detailed description of the Execution Engine platform is in Section 13.

#### 7.2.1.5.1.3.3 OCAP 1.1 Application Programming Interfaces

OCAP 1.1 contains a large set of APIs. These APIs are itemized in Section 13.

#### 7.2.1.5.1.3.4 Monitor Application

The Monitor Application is a special unbound application provided by, or created for, the MSO. The primary function of the Monitor Application is to manage the lifecycle of OCAP 1.1 applications in order to prevent harm to the cable network. The Monitor Application also centralizes basic network functions such as resource management, signaling, launching, priority, certificates, and security, in order to provide the cable consumer with multiple and diverse services in a seamless, high-quality manner.

Specific behaviors that may be governed by the Monitor Application include handling of specialized remote keys, authorizations and permissions granted to all OCAP 1.1 applications, arbitration of resource conflicts between OCAP 1.1 applications.

A detailed description of the Monitor Application is in Section 21.

### 7.2.1.5.1.4 Application Components

The Application Components are OCAP 1.1 applications that are written to run in any OCAP 1.1-compliant terminal or set-top box.

#### 7.2.1.5.1.4.1 EPG

The Electronic Program Guide (EPG) provides the user with an interface for selection of traditional TV channels as well as selected advanced services. This application is written by, or for, the cable operators and reflects the look-and-feel of their user interface to the viewer.

#### 7.2.1.5.1.4.2 VOD, Application XYZ

These applications are examples of possible OCAP 1.1 applications. VOD is the video-on-demand application. Application XYZ is a place holder for the multitude of applications that will be developed for the OCAP 1.1 host device by independent content developers.

#### 7.2.1.5.1.4.3 Non-cable Application

Non-cable applications may be written in Java and fully compatible with the OCAP EE APIs or written in Java and partly compatible with the OCAP EE APIs or written in any other language. See Section 10.2.2.7 for more information.

#### 7.2.1.5.1.4.4 'Baseline functionality

The baseline functionality is part of the OCAP 1.1 implementation. This functionality needs to be available to the viewer even if a CableCARD device is not inserted or before a Monitor Application is on-board. Without a CableCARD device, it provides clear non-encrypted channels, emergency alert system, and closed captioning. With a CableCARD device inserted, baseline functionality adds Host handling of the Host/CableCARD interface resources, CableCARD extended and data channel control, as well as Monitor Application and unbound application launching.

Baseline Functionality can be thought of as sets of related functionality that can be grouped into conceptual modules. These modules include:

- Executive - launches the Monitor Application or unbound applications,

- Watch TV - minimal remote control support and clear-to-air channel tuning,
- CableCARD Resources - Application Information, MMI, and Specific Application,
- Emergency Alert System - mandated EAS, works before and after CableCARD device insertion,
- Closed captioning - mandated closed captioning works, before and after CableCARD device insertion,
- CableCARD channels - extended and data channels initialization and control.

#### *7.2.1.5.2 Overview of the Execution Engine (EE)*

The Execution Engine consists of the Java Virtual Machine and a collection of Java APIs (the "OCAP 1.1 Java platform") that abstract the functionality of the television receiver.

##### 7.2.1.5.2.1 The OCAP 1.1 Java Platform

The OCAP 1.1 Java platform consists of Java APIs organized into multiple packages.

###### 7.2.1.5.2.1.1 Fundamental Java APIs

The OCAP 1.1 EE includes a number of existing Java APIs which were originally defined for use apart from television applications. The most significant of these are as follows:

- The fundamental Java packages of `java.lang`, `java.io`, and `java.util` provide the basis of the Java environment including the object model, text handling, multi-threading, security, a file system model, and various basic utilities.
- The `java.net` package provides access to IP based communication, both over the broadcast and via the back channel.
- The `java.awt` package provides the fundamentals required for graphical user interfaces.
- The Java Media Framework (JMF) provides an abstract representation of real-time streamed media. Various extensions are specified to tailor this to the OCAP 1.1 environment.
- The Java Secure Socket Extension ([JSSE]) allows applications to access a secure return channel connection in the same way as is done for e-commerce on the internet today.

###### 7.2.1.5.2.1.2 HAVi Level 2 User Interface APIs

The user interface API from the HAVi (Home Audio/Video Interoperability) Consortium is included in the OCAP 1.1 Java platform in order to provide support for TV specific UI functionality. This includes a number of TV-specific features including:

- a TV-oriented widget set
- extended support for remote controls as user input devices
- support for a top-level Java UI container that can reflect TV constraints
- a general framework providing support for some of the issues that need to be addressed as part of defining the relationship between video and graphics.

The `org.havi.ui` and `org.havi.ui.event` packages provide these features. The rest of the HAVi Java APIs are not required by the OCAP 1.1 Profile.

#### 7.2.1.5.2.1.3 DAVIC APIs

The OCAP 1.1 Java platform includes Java packages defined by DAVIC, the Digital Audio Visual Council. The following Java APIs from DAVIC are the primary ones included in the OCAP 1.1 Java platform:

Tuning API

Allows an application to explicitly tune the Host device. Separate network interfaces will be available for Host devices that have multiple tuners. An in-band application will not be allowed to tune away from the service or service set that it is associated with.

MPEG-2 section filter API

Allows OCAP 1.1 applications to filter data from private sections carried in a MPEG-2 transport stream.

Conditional Access API

OCAP 1.1 does not use the DAVIC Conditional Access APIs.

Streamed Media API

Provides various JMF extensions to allow control over real-time streamed television media. It adds features such as the control of subtitles and DSMCC stream events.

Content referencing API

This provides a Java encapsulation for a URL format to reference DVB network components such as transport streams, services and elementary streams.

#### 7.2.1.5.2.1.4 Java TV API

The OCAP 1.1 Java platform includes the packages defined by the Java TV API specification from Sun Microsystems. Some of the primary APIs that it defines are as follows:

- Service selection API
  Provides mechanisms for selecting, starting, stopping and monitoring the presentation of a service.
- Protocol independent SI API
  Provides an abstract API for service information that hides the nature of the underlying protocol or data format.
- JMF extensions
  For selecting individual components and controlling video positioning and scaling.
- Application lifecycle
  The Xlet interface defines an entry point by which TV applications are managed in the OCAP 1.1 Execution Engine.

#### 7.2.1.5.2.1.5 DVB-MHP APIs

The OCAP 1.1 Java platform includes the Java packages defined in the MHP specification. These include the following:

- User input event API
  Allows non-graphical applications to receive user input events.

- Persistent storage API
  Extends the file access support in `java.io` with support for various features relevant to files held in a persistent storage device such as FLASH ROM.
- User settings and preferences API
  Allows applications to query and manipulate certain standardized user preferences such as user language, country, and default font size.
- Streamed Media API
  Provides additional extensions to the Java Media Framework for DVB-specific details such as active format descriptors, and video positioning and scaling.
- Application listing and launching API
  Allows applications to obtain a list of those applications which can be launched and to launch them. Subject to security restrictions, it allows running applications to be managed as well.

#### *7.2.1.5.3 Applications and the Execution Engine*

##### 7.2.1.5.3.1 Application Model and Lifecycle

The application model for Java applications running in the Execution Engine is similar to that used by Java applets in the internet. An OCAP 1.1 Java application implements the Java TV Xlet interface which provides methods which are called by an application manager. These methods are called to start and stop the application and induce other state changes defined by a simple finite state machine.

##### 7.2.1.5.3.2 Unbound Applications

The Host device manufacturer or MSO can provide unbound OCAP applications. These applications are given higher priority so that resources are less likely to be taken from them. One example is an application that assumes Emergency Alert System message handling. The priority scheme in Table 10–2 provides for higher ranges of priorities for unbound applications. The Monitor Application is a special unbound application with the highest possible priority.

##### 7.2.1.5.3.3 Service Bound Applications

Service bound applications are associated with transport streams and are assigned priority values below the range assigned to unbound applications, (see Table 10–2). Service bound applications do not perform critical system functions and are more able to give up resources than unbound applications in the event of a resource contention.

# 8 TRANSPORT PROTOCOLS

This section contains requirements that are extensions to [DVB-GEM 1.1] Section: 6 Transport protocols and are extensions of [DVB-MHP1.1.2] Section: 6 Transport Protocols.

This section covers the transport protocols that OCAP 1.1 supports.

The OCAP 1.1 transport protocols enable an OCAP 1.1 application to communicate with the external world. The OCAP 1.1 Specification provides a way to do this through CATV network configurations and network-independent protocols.

The protocols described in this section provide a generic solution for a variety of broadcast-only and interactive services. This section covers how this is accomplished through the use of DSM-CC User-to-User Object Carousel and DSM-CC User-to-User Data Carousel protocols. It also outlines the support for IP over an OOB interaction channel, as well as over the broadcast channel.

## 8.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section. Section 8, Transport Protocols (this section) of the OCAP 1.1 Profile corresponds to Section 6 of [DVB-MHP1.1.2] and [DVB-GEM 1.1] as follows:

***Table 8–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 8Transport Protocols | 6 Transport protocols | Extension | 6 Transport Protocols | Extension |
| 8.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 8.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 8.2.1 Deviations from DVB-MHP | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 8.2.1.1 Introduction | 6.1 Introduction | Extension | 6.1 Introduction | Extension |
| 8.2.1.2 Conditional Access | 6.2 Broadcast channel protocols | Extension | 6.2 Broadcast Channel Protocols | Extension |
| 8.2.1.3 MPEG-2 Transport Stream | 6.2.1 MPEG-2 transport stream | Compliance | 6.2.1 MPEG-2 Transport Stream | Extension |
| No Corresponding Section | 6.2.2 MPEG-2 sections | Compliance | 6.2.2 MPEG-2 Sections | Compliance |
| No Corresponding Section | 6.2.3 DSM-CC private data | Compliance | 6.2.3 DSM-CC Private Data | Compliance |
| No Corresponding Section | 6.2.4 DSM-CC data carousel | Compliance | 6.2.4 DSM-CC Data Carousel | Compliance |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| No Corresponding Section | 6.2.5 Object carousel | Compliance | 6.2.5 DSM-CC User-to-User Object Carousel | Compliance |
| No Corresponding Section | 6.2.6 Protocol for delivery of IP multicast over the broadcast channel | Compliance | 6.2.6 DVB Multiprotocol Encapsulation | Compliance |
| No Corresponding Section | 6.2.7 Internet Protocol (IP) | Compliance | 6.2.7 Internet Protocol (IP) | Compliance |
| No Corresponding Section | 6.2.8 User Datagram Protocol (UDP) | Compliance | 6.2.8 User Datagram Protocol (UDP) | Compliance |
| 8.2.1.4 Service Information | 6.2.9 Service information | Extension | 6.2.9 DVB Service Information | Extension |
| No Corresponding Section | 6.2.10 IP signaling | Compliance | No Corresponding Section | |
| No Corresponding Section | 6.3 Interaction channel protocols | A subsection is extended | 6.3 Interaction Channel Protocols | A subsection is extended |
| 8.2.1.5 Network Dependent Protocols | 6.3.1 Network Dependent Protocols | Extension | 6.3.1 Network Dependent Protocols | Extension |
| No Corresponding Section | 6.3.2 Internet Protocol | Compliance | 6.3.2 Internet Protocol (IP) | Compliance |
| No Corresponding Section | 6.3.3 Transmission Control Protocol | Compliance | 6.3.3 Transmission Control Protocol (TCP) | Compliance |
| No Corresponding Section | 6.3.4 UNO-RPC | Compliance | 6.3.4 UNO-RPC | Compliance |
| No Corresponding Section | 6.3.5 UNO-CDR | Compliance | 6.3.5 UNO-CDR | Compliance |
| No Corresponding Section | 6.3.6 DSM-CC User to User 17 | Compliance | 6.3.6 DCM-CC User to User | Compliance |
| No Corresponding Section | 6.3.7 Hypertext Transfer Protocol (HTTP) | Compliance | 6.3.7 Hypertext Transfer Protocol (HTTP) | Compliance |
| No Corresponding Section | 6.3.8 Service Specific | Compliance | 6.3.8 Service Specific | Compliance |
| No Corresponding Section | 6.3.9 User Datagram Protoc18 | Compliance | 6.3.9 User Datagram Protocol (UDP) | Compliance |
| No Corresponding Section | 6.3.10 DNS | Compliance | No Corresponding Section | |
| 8.2.1.6 Multiprotocol Encapsulation | 11.5.2 Support for Multicast IP over the Broadcast Channel | Extension | 11.5.2 Support for Multicast IP over the Broadcast Channel | Extension |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 8.2.1.7 Multicast Support | 11.5.3 Support for IP over the return channel | Extension | 11.5.3 Support for IP over the return channel | Extension |
| 8.2.2 Extensions to DVB-GEM (Normative) | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 8.2.2.1 Broadcast Channel Protocols | Section 6.2 Broadcast Channel Protocols | Extension | 6.2 Broadcast Channel Protocols | Extension |
| 8.2.2.2 Interaction Channel Protocols | Section 6.3 Interaction Channel Protocols | Extension | 6.3 Interaction Channel Protocols | Extension |
| No Corresponding Section | Section 6.4 Transport protocols for application loading over the interaction channel | Compliance | Section 6.4 Transport protocols for application loading over the interaction channel | Compliance |

Section 6 of the [DVB-GEM 1.1] specification deals with the Network Independent Protocols and the networks as defined in two specifications from the DVB project, as specified in [ETS 300 802] and [EN 301 192].

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 Object Carousel and DSM-CC User-to-User Data Carousel protocols, as specified in [ISO 13818-6] and support for IP over the interaction channel as specified in [DVB-GEM 1.1] and [CCIF 2.0] and extended in this specification.

## 8.2 OCAP 1.1 Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

### 8.2.1 Deviations from DVB-MHP

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

#### *8.2.1.1 Introduction*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 6.1 Introduction and are extensions of [DVB-MHP1.1.2] Section: 6.1 Introduction.

The network-dependent protocols for the interaction channels in the OpenCable context are specified in [CCIF 2.0] and [SCTE 54] for CATV networks. The network-dependent protocols work together with the Network Independent Protocols.

#### *8.2.1.2 Conditional Access*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 6.2 Broadcast channel protocols and are extensions of [DVB-MHP1.1.2] Section: 6.2 Broadcast Channel Protocols.

The paragraph immediately following Figure 8 contains text that discusses application requests for access to CA scrambled MPEG-2 sections. This is not the case for OCAP 1.1, and this paragraph is replaced by the following text: In an OCAP environment, CA decryption is performed by the CableCARD device in a manner that is transparent to applications. The CableCARD device will initiate decryption of all authorized data without the application needing to explicitly ask for this action.

#### *8.2.1.3 MPEG-2 Transport Stream*

This subsection is compliant with [DVB-GEM 1.1] Section: 6.2.1 MPEG-2 transport stream and are extensions of [DVB-MHP1.1.2] Section: 6.2.1 MPEG-2 Transport Stream.

As applied to cable, the MPEG-2 transport stream SHALL comply with ANSI/SCTE 07 2000 [SCTE 07].

#### *8.2.1.4 Service Information*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 6.2.9 Service information and are extensions of [DVB-MHP1.1.2] Section: 6.2.9 DVB Service Information.

OCAP 1.1 supports Service Information as defined in [SCTE 65] and [SCTE 54] instead of DVB Service Information. In an OpenCable Host [SCTE 54] is used to provide Service Information in-band. In an OpenCable Host with a CableCARD device inserted, [SCTE 65] is used to provide Service Information out-of-band.

[SCTE 65] defines Service Information for application-type virtual channels and allows Host platforms to optionally support these channels. OCAP 1.1 platforms SHALL NOT expose information relating to application-type virtual channels through the Service Information interfaces described in Annex T. OCAP 1.1 platforms MAY discard Service Information relating to application-type virtual channels as defined [SCTE 65] (primarily for delivery through the Virtual Channel Map and Source Name Sub Table) and not allow this information to be retrieved by an OCAP-J application.

OCAP also supports the following descriptors defined in [ETSI EN 300 468]:

OCAP Locators that specify component_tag_elements.

transport_protocol descriptor in the AIT, with protocol_id=0x0001 ("MHP Object Carousel"). The component_tag specifies the component containing the object carousel.

Resolving association tags used in the object carousel, as defined in [DVB-MHP1.1.2] annex B.3 "AssociationTag Mapping".

- In the description of stream_identifier_descriptor in [ETSI EN 300 468], the description of the component_tag field states:

  component_tag: This 8-bit field identifies the component stream for associating it with a description given in a component descriptor. Within a program map section each stream identifier descriptor shall have a different value for this field.

- This paragraph is amended by removing the phrase "for associating it with a description given in a component descriptor". The amended description is:

  component_tag: This 8-bit field identifies the component stream. Within a program map section each stream identifier descriptor shall have a different value for this field.

**Note:** The component_descriptor is not included in OCAP 1.1.

- OCAP 1.1 complies with [DVB-MHP1.1.2] annex B.3.2 "DSM-CC association_tags to DVB component_tags".

OCAP 1.1 differs from Section 10.12 of [DVB-MHP1.1.2] in how the service identifier descriptor is delivered. In OCAP 1.1, zero or more service identifier descriptors MAY be included in the Short or Long Form Virtual Channel Table defined in [SCTE 65]. More specifically, the service identifier descriptors are carried in the descriptor section of the virtual channel record in the Short Virtual Channel Table and in the descriptor section of the inner loop of the Long-form Virtual Channel Table. Each such descriptor defines a single textual identifier for the service. The syntax of this identifier is specified in [DVB-MHP1.1.2] 14.9.1, "Syntax of the textual service identifier".

#### *8.2.1.5 Network Dependent Protocols*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 6.3.1 Network Dependent Protocols and are extensions of [DVB-MHP1.1.2] Section: 6.3.1 Network Dependent Protocols.

The network dependent protocols SHALL be as defined in ANSI/SCTE 40 [SCTE 40], etc. for CATV networks.

#### *8.2.1.6 Multiprotocol Encapsulation*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 11.5.2 Support for Multicast IP over the Broadcast Channel and are extensions of [DVB-MHP1.1.2] Section: 11.5.2 Support for Multicast IP over the Broadcast Channel.

The IP over broadcast (multi protocol Encapsulation-) channel is not supported by an OCAP device. This specification is compliant with the unsupported multi-cast in broadcast option specified by [DVB-MHP1.1.2] Section 11.5.2, item f. OCAP does not support multi-cast or uni-cast in broadcast option.

#### *8.2.1.7 Multicast Support*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 11.5.3 Support for IP over the Return Channel and are extensions of [DVB-MHP1.1.2] Section: 11.5.2 Support for IP over the Return Channel.

Where support for IP over the return channel is supported, multicast IP over the return channel is required.

### 8.2.2 Extensions to DVB-GEM (Normative)

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

The following information is normative for the OCAP 1.1 Profile. It is specific to OCAP 1.1 and is not contained within the [DVB-MHP1.1.2].

#### *8.2.2.1 Broadcast Channel Protocols*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 6.2 Broadcast Channel Protocols and are extensions of [DVB-MHP1.1.2] Section: 6.2 Broadcast Channel Protocols.

OCAP SHALL require the following additional broadcast channel protocols.

##### *8.2.2.1.1 VBI*

The OCAP 1.1 implementation SHALL support VBI as specified in [EIA-608-B], in [SCTE 21], and [SCTE 20], except non-Real Time Sampled Video related part.

#### *8.2.2.2 Interaction Channel Protocols*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 6.3 Interaction Channel Protocols and are extensions of [DVB-MHP1.1.2] Section: 6.3 Interaction Channel Protocols.

OCAP SHALL require the following interaction channel protocols.

##### *8.2.2.2.1 HDNI*

The OCAP 1.1 implementation SHALL support all transports required by [SCTE 26].

##### *8.2.2.2.2 CableCARD*

The OCAP 1.1 implementation SHALL support CableCARD/Host communication as required by [CCIF 2.0], in accordance with [CHILA].

##### *8.2.2.2.3 HTTP 1.1*

The OCAP 1.1 implementation SHALL support HTTP 1.1 over the Interaction Channel as defined in Section 7.1 of [ATSC A/96], which extends [DVB-MHP1.1.2] Section: 15 Detailed Platform Profile Definitions, Table 65 where HTTP 1.1 support is optional.

In addition to the functionality specified in Section 7.1 of [ATSC A/96], the OCAP 1.1 implementation SHALL support persistent connections as specified by Section 8.1 of HTTP 1.1, as referenced by [ATSC A/96] Section 7.1.

# 9 CONTENT FORMATS

This section contains requirements that are extensions to [DVB-GEM 1.1] Section: 7 Content formats and are extensions of [DVB-MHP1.1.2] Section: 7 Content formats.

This section of the OCAP 1.1 Profile specifies the minimal set of content formats that SHALL be supported by an OCAP 1.1-compliant implementation. It also covers the color representation for the image data. Finally, this section identifies compatible MIME types which assist an OCAP 1.1 application in identifying the type of content it is working with.

The content formats supported by the OpenCable Application Platform include:

- static image formats
- broadcast streaming formats, including audio, video, and closed-captioning
- resident fonts, as well as downloadable fonts

## 9.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section. Section 9, Content Formats (this section) of the OCAP 1.1 Profile corresponds to Section 7 of [DVB-MHP1.1.2] and [DVB-GEM 1.1] as follows:

***Table 9–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 9 Content Formats | 7 Content formats | Extension | 7 Content formats | Extension |
| 9.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 9.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 9.2.1 Deviations from the DVB-MHP Specification | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| No Corresponding Section | 7.1 Static formats | Compliance | 7.1 Static formats | A subsection is extended |
| No Corresponding Section | 7.1.1 Bitmap image formats | Compliance | 7.1.1 Bitmap image formats | Compliance |
| No Corresponding Section | 7.1.2 MPEG-2 I-Frames | Compliance | 7.1.2 MPEG-2 I-Frames | Compliance |
| No Corresponding Section | 7.1.3 MPEG-2 Video "drips" | Compliance | 7.1.3 MPEG-2 Video "drips" | Compliance |
| 9.2.1.1 Audio: Monomedia Format for Audio Clips | 7.1.4 Monomedia format for audio clips | Compliance | 7.1.4 Monomedia format for audio clips | Extension |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| No Corresponding Section | 7.1.5 Monomedia format for text | Compliance | 7.1.5 Monomedia format for text | Compliance |
| No Corresponding Section | 7.2 Broadcast streaming formats | Compliance | 7.2 Broadcast streaming formats | Compliance |
| 9.2.1.2 Audio: Broadcast Streaming Format | 7.2.1 Audio | Compliance | No Corresponding Section | |
| 9.2.1.3 Video: Broadcast Streaming Format | 7.2.2 Video | Compliance | No Corresponding Section | |
| No Corresponding Section | 7.2.3 Subtitles | Compliance | 7.2.3 Subtitles | Compliance |
| No Corresponding Section | 7.3 Resident fonts | Compliance | 7.3 Resident fonts | Compliance |
| No Corresponding Section | 7.4 Downloadable fonts | Compliance | 7.4 Downloadable fonts | Compliance |
| No Corresponding Section | 7.5 Color representation | Compliance | 7.5 Color representation | Compliance |
| 9.2.2 Extensions to DVB-GEM | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 9.2.4 MIME types | 7.6 MIME types | Extension | 7.6 MIME types | Extension |

## 9.2 OCAP 1.1 Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

### 9.2.1 Deviations from the DVB-MHP Specification

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

#### *9.2.1.1 Audio: Monomedia Format for Audio Clips*

This subsection is compliant with [DVB-GEM 1.1] Section: 7.1.4 Monomedia format for audio clips and are extensions of [DVB-MHP1.1.2] Section: 7.1.4 Monomedia format for audio clips.

This section corresponds to Section 7.1.4 of [DVB-GEM 1.1] as follows:

MPEG-1 Audio (Layers 1, 2, and 3) ES data SHALL be supported as defined in [ISO 11172-3] and constrained in [ETSI TR 101 154]. Dolby AC3 data SHALL be supported as in [ATSC A/52B] and [ATSC A/53E].

#### *9.2.1.2 Audio: Broadcast Streaming Format*

This subsection is compliant with [DVB-GEM 1.1] Section: 7.2.1 Audio.

This section corresponds to Section 7.2.1 of [DVB-GEM 1.1] as follows:

Dolby AC3 data SHALL be supported as in [ATSC A/52B] and [ATSC A/53E].

#### *9.2.1.3 Video: Broadcast Streaming Format*

This subsection is compliant with [DVB-GEM 1.1] Section: 7.2.2 Video.

This section corresponds to Section 7.2.2 of [DVB-GEM 1.1] as follows:

MPEG-2 Video with the restrictions and enhancements defined in [SCTE 43].

### 9.2.2 Extensions to DVB-GEM

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

The following information extends OCAP 1.1 Profile. It attempts to clarify issues in the [DVB-GEM 1.1] as well as provide additional information to the normative requirements.

#### *9.2.2.1 Closed-Captioning*

OCAP 1.1 SHALL support closed-captioning as specified in Section 7.2.5 of [HOST 2.0].

An application can set the preference of closed captioning text representation.

### 9.2.3 OOB/DSG Object Carousel

OCAP 1.1 SHALL support carriage of object carousel in OOB and DSG MPEG flows. Section 22.2.2.1 describes the operation of the object carousel in extended channel MPEG flows. Section 22.2.2.5 describes the operation of object carousels in DSG application tunnels. OCAP 1.1 should not expect that all networks and CableCARDs will implement support for the OOB object carousel.

### 9.2.4 MIME types

This subsection is compliant with [DVB-GEM 1.1] Section: 7.6 MIME types. This section corresponds to Section 7.6 of [DVB-GEM 1.1] as follows:

OCAP defines following MIME Types in addition to defined MIME types in Section 7.6 of [DVB-GEM 1.1]:

| MIME type | Extension | Definition of content |
|---|---|---|
| audio/ac3 | ".ac3" | As defined in 9.2.1.1 Audio: Monomedia Format for Audio Clips |

# 10 APPLICATION MODEL

This section contains requirements that are extensions to [DVB-GEM 1.1] Section: 9 Application model and are extensions of [DVB-MHP1.1.2] Section: 9 Application model.

The application model defines the control of the lifecycle of OCAP 1.1 applications. The OCAP 1.1 Profile extends the DVB-MHP service bound application model to include unbound applications.

DVB-MHP defines service bound applications as those that depend upon signaling within a broadcast service (the AIT) for execution. DVB-MHP accommodates multiple bound applications within a broadcast service and explicitly states that these applications terminate upon service change unless they are also bound to the new service. OCAP 1.1 extends the DVB-MHP application model via the introduction of unbound applications. Unbound applications and Monitor Applications are broadcast service independent. OCAP 1.1 uses the service context to present an abstract service. One or more unbound applications are associated with a single abstract service. Abstract services are a mechanism to group a set of related unbound applications where some aggregator has taken the responsibility to ensure that the set of related applications work together. This is a generalization of a broadcast service to support applications not related to a broadcast service. For example, e-mail and chat applications could be bundled together and authorized using an abstract service. Unbound applications MAY be initiated and controlled by specific API calls from another unbound application. This API is specified in.Annex G. Unbound applications MAY also be initiated and controlled via the XAIT.

## 10.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section. Section 10 (this section) of the OCAP 1.1 Profile corresponds to Section 9 of [DVB-MHP1.1.2] and [DVB-GEM 1.1] as follows:

***Table 10–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 10 Application Model | 9 Application model | Extension | 9 Application model | Extension |
| 10.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 10.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 10.2.1 Deviations from the DVB-MHP Specification | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 10.2.1.1 General | 9 Application model | Extension | 9 Application model | Extension |
| 10.2.1.2 Application Model of Bound Application | 9.1 Broadcast GEM application | Extension | 9.1 Broadcast MHP applications | Extension |
| 10.2.1.3 OCAP-J Model | 9.2 DVB-J model | Extension | 9.2 DVB-J model | Extension |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| No Corresponding Section | 9.3 DVB-HTML model | Compliance | 9.3 DVB-HTML model | Compliance |
| 10.2.1.4 Inter-application Resource Management | 9.5 Inter-application resource management | Extension | 9.4 Inter-application resource management | Extension |
| No Corresponding Section | 9.7 Services and application not related to conventional DVB services | Compliance | 9.7 Services and application not related to conventional DVB services | Compliance |
| No Corresponding Section | 9.8 Lifecycle of internet access applications | Compliance | 9.8 Lifecycle of internet access applications | Compliance |
| No Corresponding Section | 9.9 Plug-ins | Compliance | 9.9 Plug-ins | Compliance |
| No Corresponding Section | 9/10 Stored and Cached applications | Compliance | 9/10 Stored and Cached applications | Compliance |
| No Corresponding Section | 9.11 Lifecycle interactions between MHP and resident applications | Compliance | 9.11 Lifecycle interactions between MHP and resident applications | Compliance |
| 10.2.2 Extensions to DVB-GEM (Normative) | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |

## 10.2 OCAP 1.1 Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

### 10.2.1 Deviations from the DVB-MHP Specification

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

#### *10.2.1.1 General*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 9 Application model and are extensions of [DVB-MHP1.1.2] Section: 9 Application model.

All instances of the term "MHP" in the Section 9 Application model and its sub-sections of [DVB-GEM 1.1] SHALL be replaced by "OCAP".

All instances of the term "DVB-J" in the Section 9 Application model and its sub-sections of [DVB-GEM 1.1] SHALL be replaced by "OCAP-J".

All instances of the term "MHP application" in the Section 9 Application model and its sub-sections of [DVB-GEM 1.1] SHALL be replaced by "bound application".

The term "navigator" in the Section 9 Application model and its sub sections of [DVB-GEM 1.1] SHALL be replaced by "unbound application".

OCAP-J application types MAY make use of any OCAP 1.1 API as permissions allow. All applications in OCAP 1.1 SHALL be signaled as type OCAP-J.

#### *10.2.1.2 Application Model of Bound Application*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 9.1 Broadcast GEM application and are extensions of [DVB-MHP1.1.2] Section: 9.1 Broadcast MHP applications.

The bound application of OCAP 1.1 corresponds to the broadcast MHP application. OCAP 1.1 extends the application model of the bound application as defined in [DVB-GEM 1.1] Section 9.1 Broadcast MHP application and its subsections.

The term "broadcast MHP application" in the Section 9.1.1 Basic lifecycle control of [DVB-GEM 1.1] is replaced by "bound application". OCAP 1.1 does not comply with the description "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" in [DVB-MHP1.1.2], Section 9.1.1 Basic lifecycle.

#### *10.2.1.3 OCAP-J Model*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 9.2 DVB-J model and are extensions of [DVB-MHP1.1.2] Section: 9.2 DVB-J Model.

#### *10.2.1.4 Inter-application Resource Management*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 9.4 Inter-application resource management and are extensions of [DVB-MHP1.1.2] Section: 9.4 Inter application resource management.

Any application SHALL adhere to application priority and resource management as defined in Section 10 and Section 19, respectively.

### 10.2.2 Extensions to DVB-GEM (Normative)

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

The following information is normative for the OCAP 1.1 Profile. It is specific to OCAP 1.1 and is not contained within the [DVB-MHP1.1.2].

#### *10.2.2.1 OCAP Applications*

In [DVB-MHP1.1.2] all applications are related to a service and their lifecycle is linked to the currently selected broadcast service. OCAP 1.1 introduces applications that are not linked to the broadcast service and whose lifecycle is independent of the AIT signaling. OCAP 1.1 extends the service model by introducing the concept of abstract services that contain applications from the following sources:

a. Host device manufacturer applications installed with the OCAP 1.1 implementation

b. Applications that are signaled through the XAIT

c. Applications that are registered through the Monitor Application

These three types of applications are known generically as unbound applications.

OCAP further extends the MHP application model with the concept of applications having modes. These modes are defined in 10.2.2.8 below.

This section describes the extensions to the [DVB-GEM 1.1] broadcast service and application lifecycle model that are available in OCAP 1.1 to manage these types of applications.

#### *10.2.2.2 OCAP Service Model*

##### *10.2.2.2.1 Service Context Creation and Destruction*

The OCAP 1.1 Profile defines two types of service, a broadcast service and an abstract service. Each of these types of service run within a service context and service contexts are created by the implementation as follows:

a. The implementation creates a service context for each concurrent broadcast service that it can support presenting simultaneously. A specific broadcast service is associated with the service context through service selection. The number of concurrently presenting broadcast services is implementation dependent, but at least one service context SHALL always be provided to present broadcast services.

b. Each host device manufacturer application contains a service_id that defines the service with which it is associated. The set of abstract services needed for host device manufacturer applications is known at boot time and the implementation SHALL create a service context for each of these abstract services that are required by the manufacturer to be started in the boot process. All host device manufacturer applications SHALL be associated with an abstract service.

c. The implementation creates a service context for each set of applications signaled through the XAIT or registered by an application with monitorAppPermission("registrar") that are to be associated with a separate abstract service that is signaled as "auto_select = true". The number of service contexts that are created for these applications varies over time (for example, when a new application is signaled in the XAIT with a new abstract service identification) and the implementation needs to create service contexts for these abstract services as needed. The service is defined by the service_id in the same manner as for host device manufacturer applications.

d. Applications with permission to create new service contexts MAY request the implementation to create a new context. These service contexts MAY be used to present services that are signaled as "auto_select = false". Calls to ServiceContextFactory.createServiceContext () by an application with monitorAppPermission ("servicemanager") should always return a new ServiceContext if the system has sufficient memory to allow the creation of a new service context.

It is implementation-dependent whether a service context that has been associated with an abstract service and is now in the "not presenting" state is destroyed by the implementation. When a service context is destroyed by the implementation, a ServiceContextDestroyedEvent is raised, and any application that has registered a ServiceContextListener receives this event.

##### *10.2.2.2.2 Abstract Services*

The abstract service provides a means to group co-operating unbound applications into a logical unit for presentation to the viewer. For example, a games suite may consist of a games portal application and a series of individual game applications. There are three means through which abstract services may be created:

a. The abstract service is defined in the XAIT signaling

b. The abstract service is defined in the XAIT parameter used in a call to registerUnboundApp()

c. The abstract service is associated with a host device manufacturer application

The set of applications associated with abstract services defined by (1) and (2) may vary over time as new signaling is received or new calls to registerUnboundApp() occur. Both of these forms of abstract service content definition may be used to group applications into the same abstract service. The set of applications associated with abstract services defined by (3) is immutable between boot processes; thus applications signaled in the XAIT or defined in a XAIT parameter cannot alter host device manufacturer abstract services.

Abstract services are identified by its unique service_id and abstract service information is stored in the services database. When an abstract service is selected, the information relating to the applications within that service is stored in the applications database. When an abstract service is terminated, the information relating to the applications within that service is removed from the applications database.

New abstract services SHALL be created in the services database by signaling new service identifiers in the XAIT or by calling registerUnboundApp() with a XAIT fragment that contains one or more new service identifiers in abstract service descriptors. Abstract services SHALL be deleted from the services database once no applications are associated with the service. This can occur either by calling unregisterUnboundApp() on each registered application, or by calling registerUnboundApp() with a XAIT fragment that lists the service with no applications associated with that service identifier, or by removing the abstract service and/or its associated applications from the XAIT signaling.

The creation and deletion of abstract services reflects the current state of applications signaled in the XAIT and applications registered or unregistered by the Monitor Application. An application that creates a new abstract service using registerUnboundApp() SHALL also register at least one unbound application to that service in the same method call. The set of services associated with host device manufacturer applications remains static until the next implementation download.

The services database SHALL provide access to a list of all applications associated with a particular abstract service through the org.ocap.service AbstractService.getAppIDs() and the AbstractService.getAppAttributes() methods. For applications signaled in the XAIT this MAY be through reference to the current state of XAIT data. For applications that have been registered by a Monitor Application, this SHALL include a copy of the information provided through the registerUnboundApp() method and this information SHALL be retained until unregisterUnboundApp() is called for this application or until the abstract service is removed. For services associated with a host device manufacturer application the source of the information is implementation defined.

When a new version of the XAIT is received the implementation SHALL update the SI database to reflect changes in the XAIT, as follows:

a. When an abstract service descriptor is no longer signaled in the XAIT and the abstract service is not currently selected, the abstract service is removed from the list of available abstract services. A ServiceDetailsChangeEvent of type REMOVE is generated.

b. When an abstract service descriptor is no longer signaled in the XAIT and the service is currently selected, the abstract service is marked for removal and will be removed when presentation of the abstract service is stopped. A ServiceDetailsChangeEvent of type REMOVE is generated when the abstract service is marked for removal and the abstract service is no longer available for selection. Any attempt to select an abstract service that is marked for removal results in an InvalidServiceComponentException from the ServiceContext.select() method.

c. When an abstract service descriptor is newly signaled in the XAIT, the abstract service is added to the list of available abstract services. A ServiceDetailsChangeEvent of type ADD is generated. The auto_select flag determines whether the abstract service is immediately selected, as described in Section 10.2.2.2.1.

d. When the details of the abstract service or its associated applications have changed, the SI database is modified to reflect the current XAIT signaling information. A ServiceDetailsChangeEvent of type MODIFY is generated.

The management of XAIT signaled applications associated with selected services is described in Section 10.2.2.3.1. The management of application storage for XAIT signaled applications is described in Section 12.2.3.

When new service and application information is provided through a call to registerUnboundApp() the OCAP implementation SHALL update the SI database to reflect changes in the XAIT as follows:

a. When an abstract service descriptor is included in the XAIT parameter without any associated applications and the abstract service is not currently selected, the applications previously registered through registerUnboundApp() for this abstract service are deleted. If no applications (including those signaled in the XAIT) remain associated with this abstract service, then the abstract service is removed from the list of available abstract service and a ServiceDetailsChangeEvent of type REMOVE is generated.

b. When an abstract service descriptor is included in the XAIT parameter without any associated applications and the abstract service is currently selected, the applications previously registered through registerUnboundApp() for this abstract service are deleted (see Section 10.2.2.3.2). If no applications (including those signaled in the XAIT) remain associated with this abstract service, then the abstract service is marked for removal and the abstract service will be removed when the presentation of the abstract service is stopped. A ServiceDetailsChangeEvent of type REMOVE is generated when the b) abstract service is marked for and the abstract service is no longer available for selection. Any attempt to select an abstract service that is marked for removal results in an InvalidServiceComponentException from the ServiceContext.select() method.

c. When a abstract service descriptor is included in the XAIT parameter for a new abstract service, then the abstract service is added to the list of available abstract services. A ServiceDetailsChangeEvent of type ADD is generated. The auto_select flag determines whether the abstract service is immediately selected as described in Section 10.2.2.2.1.

d. When an abstract service descriptor is included in the XAIT parameter and the details of the abstract service or its associated applications have changed, the SI database is modified to reflect the current details in the XAIT fragment. A ServiceDetailsChangeEvent of type MODIFY is generated.

e. When an application descriptor is included in the XAIT parameter that references an abstract service that is currently listed in the SI database and the corresponding abstract service descriptor is not included in the XAIT, the SI database is modified to reflect the current details of applications associated with this abstract service in the XAIT fragment. Applications associated with this abstract service in the SI database that are not included in the XAIT are unaffected by this call to registerUnboundApp(). A ServiceDetailsChangeEvent of type MODIFY is generated.

f. When an application descriptor is included in the XAIT parameter that does not reference an abstract service that is listed in the SI database and the corresponding abstract service descriptor is not listed in the XAIT, then the application descriptor is ignored.

**Note:** The OCAP specification allows the same abstract service to be signaled through the XAIT and to be updated by a XAIT fragment supplied to registerUnboundApp(). The XAIT signaling is designed to be repetitive while registerUnboundApp() is designed as a single instance method call. The semantics of these two operations are slightly different and this can cause indeterminate results if the XAIT is changed when registerUnboundApp() is called. Operators wishing to update the same abstract service from both XAIT and registerUnboundApp() SHOULD monitor the content of the SI database after each call to verify that the requested changes have occurred.

All abstract services SHALL have their service type set to OCAP_ABSTRACT_SERVICE and applications can create a service type filter to list only the services of this type (see ServiceList.filterServices(ServiceFilter filter) in the javax.tv.service.navigation package).

The javax.tv.service.transport.ServiceDetailsChangeEvent is notified via a javax.tv.service.transport.ServiceDetailsChangeListener that is added to a concrete

javax.tv.service.transport.Transport object. If the ServiceDetailsChangeEvent represents a change of an abstract service, a javax.tv.service.navigation.ServiceDetails object returned by a ServiceDetailsChangeEvent.getServiceDetails() method shall behave as described in Annex T.2.2.16.

10.2.2.2.2.1 Examples of Abstract Service Updating

This section provides two examples to show the difference of abstract service updating between XAIT signaling and the registerUnboundApp() method.

**Note:** Some detailed parameters are omitted in the following XAIT example.

**Example 1:**

XAIT_1

```
Application_information_section {
    Table_id = 0x74
    ...
    common_descriptors_length = ***

        abstract_service_descriptor {
            descriptor_tag = 0x66
            ...
            service_id = 0x020001
            ...
            service_name_byte = service1
        }

        abstract_service_descriptor_{
            descriptor_tag = 0x66
            ...
            service_id = 0x020002
            ...
            service_name_byte = service2
        }

    Reserved_future_use
    application_loop_length = ***

        application_identifier {
            organization_id = 1
            application_id = 1
        }
        ...
        application_descriptor_loop_length = ***
            unbound_application_descriptor {
                descriptor_tag = 0x67
                ...
                service_id = 0x020001
                version_number = 0
            }
```

```
            application_identifier {
                organization_id = 1
                application_id = 2
            }
            ...
            application_descriptor_loop_length = ***
                unbound_application_descriptor {
                    descriptor_tag = 0x67
                    ...
                    service_id = 0x020002
                    version_number = 0
                }
    }
```

XAIT_2

```
    Application_information_section {
        Table_id = 0x74
        ...
        common_descriptors_length = ***
            abstract_service_descriptor {
                descriptor_tag = 0x66
                ...
                service_id = 0x020001
                ...
                service_name_byte = service1
            }
            abstract_service_descriptor_{
                descriptor_tag = 0x66
                ...
                service_id = 0x020002
                ...
                service_name_byte = service2
            }
        Reserved_future_use
        application_loop_length = ***
            application_identifier {
                organization_id = 1
                application_id = 3
            }
            ...
            application_descriptor_loop_length = ***
                unbound_application_descriptor {
                    descriptor_tag = 0x67
                    ...
                    service_id = 0x020002
                    version_number = 0
                }
    }
```

In case of XAIT signaling:

Step 1. In the initial status, there is no abstract service.

Step 2. If XAIT_1 is signaled via OOB, the following abstract services associated to applications are created.

| Services | Associated Applications |
|---|---|
| service1 (service_id = 0x020001) | {application_id = 1, organization_id = 1} |
| service2 (service_id = 0x020002) | {application_id = 2, organization_id = 1} |

Step 3. And then if XAIT_2 is signaled via OOB, abstract services associated to applications are updated with the following. Note that service1 is deleted since it has no application.

| Services | Associated Applications |
|---|---|
| service2 (service_id = 0x020002) | {application_id = 3, organization_id = 1} |

In case of calling registerUnboundApp():

Step 1. In the initial status, there is no abstract service.

Step 2. If registerUnboundApp(XAIT_1) is called, the following abstract services associated to applications are created.

| Services | Associated Applications |
|---|---|
| service1 (service_id = 0x020001) | {application_id = 1, organization_id = 1} |
| service2 (service_id = 0x020002) | {application_id = 2, organization_id = 1} |

Step 3. And then if registerUnboundApp(XAIT_2) is called, abstract services associated to applications are updated with the following. Note that service1 is deleted since it has no application.

| Services | Associated Applications |
|---|---|
| service2 (service_id = 0x020002) | {application_id = 3, organization_id = 1} |

Example 2:

XAIT_1

```
Application_information_section {
    Table_id = 0x74
    ...
    common_descriptors_length = ***

        abstract_service_descriptor {
            descriptor_tag = 0x66
            ...
            service_id = 0x020001
            ...
            service_name_byte = service1
        }
```

```
            abstract_service_descriptor_{
                descriptor_tag = 0x66
                ...
                service_id = 0x020002
                ...
                service_name_byte = service2
            }

        Reserved_future_use
        application_loop_length = ***

            application_identifier {
                organization_id = 1
                application_id = 1
            }
            ...
            application_descriptor_loop_length = ***
                unbound_application_descriptor {
                    descriptor_tag = 0x67
                    ...
                    service_id = 0x020001
                    version_number = 0
                }

            application_identifier {
                organization_id = 1
                application_id = 2
            }
            ...
            application_descriptor_loop_length = ***
                unbound_application_descriptor {
                    descriptor_tag = 0x67
                    ...
                    service_id = 0x020002
                    version_number = 0
                }

    }
```

XAIT_2

```
    Application_information_section {

        Table_id = 0x74
        ...
        common_descriptors_length = ***

        Reserved_future_use
        application_loop_length = ***
```

```
            application_identifier {
                organization_id = 1
                application_id = 3
            }
            ...
            application_descriptor_loop_length = ***
                unbound_application_descriptor {
                    descriptor_tag = 0x67
                    ...
                    service_id = 0x020002
                    version_number = 0
                }
    }
```

In case of XAIT signaling:

Step 1. In the initial status, there is no abstract service in the service database.

Step 2. If XAIT_1 is signaled via OOB, the following abstract services associated to applications are created.

| Services | Associated Applications |
|---|---|
| service1 (service_id = 0x020001) | {application_id = 1, organization_id = 1} |
| service2 (service_id = 0x020002) | {application_id = 2, organization_id = 1} |

Step 3. And then if XAIT_2 is signaled via OOB, no abstract services exists in the service database since there is no abstract_service_descriptor.

In case of calling registerUnboundApp():

Step 1. In the initial status, there is no abstract service.

Step 2. If registerUnboundApp(XAIT_1) is called, the following abstract services associated to applications are created.

| Services | Associated Applications |
|---|---|
| service1 (service_id = 0x020001) | {application_id = 1, organization_id = 1} |
| service2 (service_id = 0x020002) | {application_id = 2, organization_id = 1} |

Step 3. And then, if registerUnboundApp(XAIT_2) is called, abstract services associated to applications are updated with the following:

| Services | Associated Applications |
|---|---|
| service1 (service_id = 0x020001) | {application_id = 1, organization_id = 1} |
| service2 (service_id = 0x020002) | {application_id = 2, organization_id = 1}<br>{application_id = 3, organization_id = 1} |

#### 10.2.2.2.2.2 Abstract Service Selection

Where an abstract service is signaled as "auto_select" this abstract service SHALL be presented in a new service context during the boot process or, if the abstract service is created after the boot process has completed, as soon as the abstract service is listed in the services database.

Only one concurrent instance of an abstract service may be selected. Any attempt by an application to select a currently presenting abstract service SHALL cause an org.javax.tv.service.selection. SelectionFailedEvent to be generated.

The OCAP 1.1 implementation SHALL allow applications within an abstract service to be executed and presented concurrently as indicated by the application signaling. Additionally, the OCAP 1.1 implementation SHALL allow both broadcast services and abstract services to execute concurrently, subject to the constraints of available resources in which case the behavior is determined by application priority and resource management.

When an abstract service is selected to be presented in either a current or newly created service context, the applications in this abstract service are entered into the applications database. The application lifecycle model for applications running within an abstract service follows the same rules as for broadcast services. Thus, all AUTO_START applications associated with an abstract service are launched, subject to filtering, when the service is selected and all applications associated with an abstract service are destroyed when the service is stopped. Unlike the broadcast service, an abstract service does not have the concept of an application that survives service termination and all applications that are signaled in the XAIT are "service bound" applications. In this context the term "service bound" has the meaning specified in [DVB-MHP1.1.2] Section 9.1.4.1 and the org.ocap.application.OcapAppAttributes.getIsServiceBound() method always returns "true" for applications running in an abstract service.

The abstract service represents a set of unbound applications that are associated with it. In the case that no applications in a currently selected abstract service are signaled as AUTO_START or PRESENT, the abstract service is considered to have terminated. The implementation SHOULD release all resources associated with the abstract service on termination and SHALL remove all references to this abstract service from the services database. The implementation SHALL be able to create a separate service context for each of the abstract services that it needs to support. A return value of AbstractServiceType.OCAP_ABSTRACT_SERVICE from AbstractService.getServiceType() is used to indicate an abstract service.

### *10.2.2.2.3 Service Context Permission, Select Permission, and Service Type Permission*

The [Java TV] specification defines a ServiceContextPermission that limits an application's control over the service context lifecycle and a SelectPermission that limits an applications ability to present a particular service in a service context. This specification defines the org.ocap.service.ServiceTypePermission which limits an application's ability to present services belonging to specific service types in a service context; see Annex P. In addition to the default ServiceTypePermission given in this section, an application MAY have ServiceTypePermission granted or revoked via the permission request file; see Section 14.2.2.1.1.

In this specification, a caller's "own" service contexts are the ServiceContext that would be returned were it to call ServiceContextFactory.getServiceContext(java.tv.xlet.XletContext) for its own XletContext, plus any service contexts that it has created through ServiceContextFactory.createServiceContext().

#### 10.2.2.2.3.1 Unsigned Application Permissions

An unsigned application has the ServiceContextPermission specified in [DVB-MHP1.1.2], section 11.10.1.9. An unsigned application does not have SelectPermission or ServiceTypePermission for any locator or action string.

#### 10.2.2.2.3.2 Signed Application Permissions

A signed application has the following set of permissions unless denied by an entry in the permission request file. The following set of permissions is defined in javax.tv.service.selection.ServiceContextPermission:

- ServiceContextPermission("*","own") to allow it to manage its own service context.

The following set of permissions is defined in org.ocap.service.ServiceTypePermission:

- ServiceTypePermission("broadcast", "own") for bound applications.
- ServiceTypePermission ("abstract.mso", "own") for unbound network applications.
- ServiceTypePermission("abstract.manufacturer", "own") for unbound manufacturer applications.

The SelectPermission assigned to a signed application is that implied by the ServiceTypePermission for the corresponding application type, unless otherwise denied by an entry in the Permission Request File. No additional SelectPermission is assigned. If SelectPermission is denied by an entry in the Permission Request File, the permissions assigned are specified for an unsigned application in Section 10.2.2.2.3.1 and the ServiceTypePermission settings in the Permission Request File are ignored.

#### 10.2.2.2.3.3 Monitor Application Permissions

An application with monitorAppPermission("service") permission has the following set of permissions defined in javax.tv.service.selection.ServiceContextPermission:

- ServiceContextPermission("access", "*") to allow it to access other service contexts
- ServiceContextPermission("getServiceContentHandlers", "own") to allow it to access its own JMF Player(s)
- ServiceContextPermission("create", "own") to allow it to create new service contexts
- ServiceContextPermission("destroy", "own") to allow it to destroy service contexts that it has created
- ServiceContextPermission("stop", "*") to allow it to cause any another abstract service to stop presenting.

The following set of permissions as defined in javax.tv.service.selection.SelectPermission:

- SelectPermission("*", "own") to allow it to select any service to start presenting in a service context that it has created.

The following set of permissions defined in org.ocap.service.ServiceTypePermission.

- The set of permissions designated by the ServiceTypePermission entry in the Permission Request File or, by default, ServiceTypePermission("*", "own").

This set of permissions allows a signed application to create a new service context and to manage the lifecycle of that service context. It also allows an application to stop a service that is currently presenting in another service context, rather than allowing the implementation to choose which service to stop.

If SelectPermission is denied by an entry in the Permission Request File, then permissions assigned are as specified for an unsigned application in Section 10.2.2.2.3.1 and the ServiceTypePermission settings in the Permission Request File are ignored.

An application with monitorAppPermission("servicemanager") permissions has the following set of permissions as defined in javax.tv.service.selection.ServiceContextPermission:

- ServiceContextPermission("access", "*") to allow it to access other service contexts.

- ServiceContextPermission("getServiceContentHandlers", "own") to allow it to access its own JMF Player(s).
- ServiceContextPermission("create", "own") to allow it to create new service contexts.
- ServiceContextPermission("destroy", "own") to allow it to destroy service contexts that it has created.
- ServiceContextPermission("stop", "*") to allow it to cause any another abstract service to stop presenting.

The following set of permissions as defined in javax.tv.service.selection.SelectPermission:

- SelectPermission("*", "own") to allow it to select any service to start presenting in a service context that it has created.

The following set of permissions as defined in org.ocap.service.ServiceTypePermission:

- the set of permissions designated by the ServiceTypePermission entry in the Permission Request File or, by default, ServiceTypePermission("*", "*")

If SelectPermission is denied by an entry in the Permission Request File, then permissions assigned are as specified for an unsigned application in Section 10.2.2.2.3.1 and the ServiceTypePermission settings in the Permission Request File are ignored.

In addition an application with MonitorAppPermission( "servicemanager") has life cycle control permission to control the life cycles of applications in other service contexts as mentioned in Section 10.2.2.3. An applications with MonitorAppPermission("servicemanager") also has AppsControlPermission as defined in [DVB-MHP1.1.2] to control the life cycles of applications in its own service context.

#### *10.2.2.2.4 Broadcast Service Selection*

The [DVB-MHP1.1.2] specification allows a signed application to select the service that is to be presented in its "own" service context. This permission is available to applications running in either a broadcast service or an abstract service.

OCAP 1.1 is fully compliant with [DVB-MHP1.1.2] (in particular Section 9.1.1 Basic lifecycle control) with respect to broadcast services. An application running in a broadcast service is prohibited from selecting a service to be presented in any other service context than its own - there is no permission provided to enable such selection. Likewise an application running in a broadcast service is prohibited from creating a new service context.

Using the permissions specified in Section 10.2.2.2.3.3, an application with monitorAppPermission("service") permission is able to select a broadcast service to present in any service context in the category "own". Such an application is also able to "stop" other broadcast services from presenting in other service contexts.

Note that the behavior of applications in the case that the same broadcast service is presented in two service contexts is undefined. The problems that arise through resource contention from the same application in the same broadcast service in two separate service contexts are not defined in [DVB-MHP1.1.2] and this is the subject of an unresolved issue against that specification.

### *10.2.2.3 Application Signaling and Lifecycle*

The OCAP-J application lifecycle differs from that described in [DVB-MHP1.1.2] in the following way:

The term "application" in Section 9.2.3 of [DVB-MHP1.1.2] is replaced by the term "application instance". The lifecycle of an OCAP-J application and an OCAP-J application instance are the same except when an OCAP-J application instance is destroyed. An OCAP-J application instance is only transiently destroyed and then moves to the unloaded state (See org.dvb.application.AppProxy.NOT_LOADED in Annex S of [DVB-MHP1.1.2]). In the case that the OCAP-J application is signaled as AUTO_START, the rules described in Section 9.1.6 of [DVB-

MHP1.1.2] apply and a new OCAP-J application instance is not auto-started when the application returns to the unloaded state.

These extensions apply to all OCAP-J applications irrespective of their source.

An application that has monitorAppPermission("servicemanager") permission has lifecycle control over all applications currently listed in the applications database. Any application that is granted this permission should only use the lifecycle control capability with caution; the normal way to control applications is by stopping the service in which they run rather than by stopping individual applications.

For an application with MonitorAppPermission("servicemanager") to be able to modify the life-cycle of any application, it SHALL be able to access the org.dvb.application.AppProxy of any application. This requires an extension to the [DVB-MHP1.1.2] definition of a "currently available" application. OCAP extends the definition as follows; when an application is granted MonitorAppPermission("servicemanager"), a "currently available" application is any application signaled from the network or registered via API to, or by the implementation. Thus, when such an application calls org.dvb.application.AppsDatabase.getInstance, a database will be populated with all of the applications that fit this definition without regard for the visibility field. In addition, OCAP extends the definition of "the current service" as follows; for applications with MonitorAppPermission("servicemanager") only, references to "the current service" as found in the [DVB-MHP1.1.2], org.dvb.application. CurrentServiceFilter and RunningApplicationsFilter class descriptions SHALL be read as "any current service".

An application that has MonitorAppPermission("testApplications") can control the handling of applications signaled with the "test_application_flag" set in their XAIT/AIT (see [DVB-MHP1.1.2] section 10.4.6). The method AppManagerProxy.enableTestApplications() may be used to configure the implementation such that applications signaled with this bit set to 1 are treated as if this field is set to zero.

#### *10.2.2.3.1 Applications Signaled in the XAIT*

Applications may be signaled through the XAIT transmitted in an Extended Channel MPEG section flow. The information in the XAIT contains all of the necessary details to create an entry in the Applications Database for each application signaled in a selected abstract service. When a new version of the XAIT is received the implementation SHALL update the Application Database with the currently signaled information for each application associated with a selected service as follows:

a. When an application does not currently exist in the Application Database the implementation SHALL create an entry for the application which contains the information in the XAIT signaling. The life-cycle control rules for XAIT signaled applications entered into the Application Database are the same as for broadcast applications signaled in AIT.

b. When an application currently exists in the Application Database that was previously signaled in the XAIT but is no longer signaled, then the implementation SHALL destroy any active application instance and remove the application from the Application Database.

c. When an application currently exists in the Application Database with the same version number as that signaled in the XAIT, the entry in the Application Database is updated to reflect the currently signaled information. Changes in the application control code are managed in the same manner as for broadcast applications signaled in the AIT.

d. When an application currently exists in the Application Database and a newer version number of the application is signaled in the XAIT then, for an entry that is not associated with an active application instance, the Application Database is updated to reflect the currently signaled information. When an application instance is launched the new version number SHALL be used.

e. When an application currently exists in the Application Database and a newer version number of the application is signaled in the XAIT then, for an entry that is associated with an active application instance,

the implementation SHALL:

f. Complete processing of XAIT signaling for older versions of the same application. This may cause the previous version of the application instance to be destroyed through a change of its application control code, in which case item (d) above applies.

g. The rules for creating an application instance only allow a single concurrently executing version of an application, therefore if the previous version of the application is currently executing, the new version of the application SHALL NOT be started by the implementation. The Application Database for the previous entry is updated to indicate that a new version of the application is signaled (see OCAPAppAttributes.hasNewVersion()) and the implementation SHALL NOT modify the Application Database to reflect the details of the new version. The implementation SHALL create an AppsDatabaseEvent to indicate that the new version of this application is available.

In the case of (e)(2) above, it is not the responsibility of the implementation to manage the transition between old and new versions of a XAIT signaled application. When the older version of the application is destroyed and no current application instance remains, the implementation SHALL update the Application Database to replace the details of the older version with those of the newer version and SHALL manage the introduction of the newer version as for a newly introduced application.

**Note:** The XAIT may simultaneously signal an older and a newer version of the same application. This can be used when an emergency update of the application is needed. The XAIT entry for the older version can destroy the application instance through the use of the DESTROY application control code so that an application instance of the newer version can be created. This may present a confusing experience to the consumer, but may be required when an application is exhibiting unwanted behavior.

All changes in the XAIT file which result in an Application Database change cause an AppsDatabaseEvent to occur and the Monitor Application MAY listen for this event in order to monitor database changes that occur to selected abstract services as a result of a new version of the XAIT.

Changes in an application's control code in the XAIT are reflected in the lifecycle of the application in the same manner as for bound applications. Thus the AUTO_START control code can be used to start applications and the PRESENT control code to indicate the availability of an application in the abstract service. As with bound applications, all AUTO_START applications are launched on service selection and any application later signaled as AUTO_START SHALL be launched by the implementation in the signaled abstract service. The effect of the use of a DESTROY or KILL control code SHOULD be considered before including these in the signaling as these codes will cause the application's Xlet.destroyXlet() method to be called and the application instance to terminate.

The XAIT also contains details to manage application storage. The implementation updates persistent application storage as described in Section 12.

As a special case, the Monitor Application is the first application started during the boot sequence and this application is associated with its own abstract service. The initial Monitor Application SHALL be the only AUTO_START application in this abstract service and is identified by having an application priority of 255. This ensures that the initial Monitor Application has the opportunity to start up before other unbound applications are launched.

#### *10.2.2.3.2 Applications registered by a privileged application*

An application with monitorAppPermission("registrar") permission MAY register applications to previously created abstract services or may create new abstract services. When an abstract service includes a registered application that service SHALL remain in the list of available services until all registered applications in that service are unregistered. The application lifecycle for these applications is managed through the registration process and is similar to that described in the previous section for Applications Signaled in the XAIT.

The implementation SHALL apply the following rules when managing applications through calls to registerUnboundApp():

a. When an application that is currently signaled in the XAIT or an application that is associated with a host manufacturer abstract service is included in the XAIT fragment, then this application entry in the XAIT is ignored.

b. Other entries in the Applications Database for currently selected abstract services are updated in the manner defined in Section 10.2.2.3.1.

Attempts to use AppManagerProxy.unregisterUnboundApp() to remove entries for applications signaled in the XAIT results in an InvalidArgumentException.

Similarly, the XAIT signaling SHALL NOT be used to change an application that was originally registered using AppManagerProxy.registerUnboundApp(). Any part of the XAIT content that refers to a registered application SHALL be ignored; all other parts of the XAIT content that do not refer to registered applications SHALL still be acted upon.

#### *10.2.2.4 Environments*

OCAP host devices SHALL support a cable environment as described in the specification for the class org.ocap.envrionment.Environment. OCAP host devices MAY also support one or more non-cable environments. OCAP host devices which only support a cable environment SHALL have that environment always in the selected state. On such hosts, application modes SHALL not be implemented and hence the application_mode_descriptor SHALL be ignored.

All application instances SHALL have a home environment which is assigned when the application instance starts and which cannot be changed for the lifetime of that application instance.

OCAP host devices which support a cable environment and a manufacturer environment SHOULD include at least the following combinations of environments and states;

***Table 10–2 - Combinations of environments and states***

| Condition | Manufacturer environment | Cable environment |
|---|---|---|
| no CableCARD inserted | selected | inactive |
| end-user is interacting with OCAP applications | background | selected |
| end-user is interacting with non-OCAP applications | selected | background |
| PiP/PoP is in effect and PiP primary event screen contains OCAP applications | presenting | selected |
| PiP/PoP is in effect and PiP primary event screen contains non-OCAP applications | selected | presenting |

When an environment is connected to multiple PiP/PoP sessions and one of these is the PiP primary event screen then that environment SHALL be in the selected state.

When a host boots, the first environment selected is addressed in Section 20.2.2.

##### *10.2.2.4.1 Initiation of state transitions*

Environments become selected as a consequence of choices made by the end user.

- Some hosts may permit the end-user to choose between cable and non-cable as two separate and distinct worlds. On such hosts, it will be obvious to the end-user when the cable and non-cable environments are selected. (It would be up to the implementation of each environment to decide which application within the environment was then started).
- Some hosts may have specific remote control buttons bound to specific applications. While the most common use of these is utility applications (e.g., audio volume control), their use with other applications is not excluded. In which case, the end user pressing one of these buttons would result in the application concerned being started with its home environment selected.
- Some hosts may permit the end-user to choose specific cable or non-cable applications using some kind of application manager UI. Normally choosing a cable application results in the cable environment becoming selected and vice-versa for choosing a non-cable application. The exception to this is choosing an application which runs in cross-environment mode. Choosing such an application does not cause any change in the selected environment.
- An application running in cross-environment mode may show a user-interface offering the end-user a choice of continuing or dismissing that application. If the end-user decides to continue, this may the application to change the selected environment in order for it to run properly. An application running in background mode may request a change of selected environment at any time.
- NOTE: Cross environment and background applications should only request an environment change when absolutely needed. For example if an application cannot get a resource due to the policy of the currently selected environment.
- The end-user may have defined that connecting certain peripherals or media to an host should automatically start an application to handle that content. Such applications could be cable applications (e.g., a photo viewer) or non-cable applications (e.g., photo-viewer, DVD player, camcorder player). Connecting that peripheral or media is considered to be the end-user choosing the environment needed by the end-users chosen application.
- Those hosts which support PIP or POP may have one PIP session showing cable content / applications and a second PIP session showing non-cable content / applications. Under these circumstances, if the end-user changes the PIP primary event screen between the two PIP sessions, that will change the selected environment accordingly.

#### *10.2.2.5 Non-cable applications*

This class of applications is an OCAP 1.1 extension to the [DVB-MHP1.1.2]. These applications are delivered with the OCAP 1.1 compliant host device and MAY be updated with each new download image of the OCAP 1.1 implementation.

##### *10.2.2.5.1 Introduction (informative)*

Non-cable applications may be OCAP applications or non-OCAP applications. A non-cable application that is not an OCAP application can never access the embedded cable return channel and is limited to functioning as a cross-environment or background application when the cable environment is selected. Implementers of non-cable applications may of course partition what is logically a single application into two pieces: an OCAP application containing all access to the embedded cable return channel and a non-OCAP application containing the rest of the logic.

Non-cable applications may be written in any language including (but not limited to) C, C++ and Java. Having non-cable applications written in Java and MSO applications written in Java running in the same Java virtual machine instance is not explicitly excluded and is a perfectly valid implementation choice as long all requirements of the OCAP specification are complied with.

Non-cable applications written in Java may call the OCAP APIs. In this situation, it is possible that for particular API calls, the OCAP API implementation may check whether the caller is a non-cable application and have different results from that defined in the OCAP specification. For example, the method NetworkInterfaceManager.getNetworkInterfaces() may return an array including an 8-VSB terrestrial tuner only when called by a manufacturer Java application. Clearly OCAP applications must see a consistent view of the platform in which they are running. Hence, in the previous example, they would not receive NetworkInterfaceEvents relating to the 8-VSB tuner only visible to manufacturer Java applications.

The above is also applicable to non-cable, non-manufacturer applications and environments, e.g., a Blu-ray environment which runs BD-J applications. A method in both the Blu-ray and OCAP specifications may have a single implementation which detects the type of the calling application and complies with the appropriate specification for that application.

***Table 10–3 - Possible combinations of cable, non-cable and OCAP applications***

| | Cable Application | Non-Cable Application |
|---|---|---|
| OCAP application | Typical MSO unbound application<br>Bound applications | Host Device Manufacture Applications (i.e., non-cable applications listed in the applications database) |
| Non-OCAP application | Void | Non-cable applications not listed in the applications database |

Some applications may be able to run both as OCAP applications and as non-OCAP applications. Examples of these include;

- service bound applications compliant with the OCAP specification and associated with services that can be presented by a non-cable environment. These run as OCAP applications when the associated service is presented by the cable environment. They MAY run as non-OCAP applications when the associated service is presented by a non-cable environment, as when the device is presenting cable video services as would a UDCR.
- Non-cable applications MAY run as OCAP applications when presented by the cable environment and as non-OCAP applications when presented by a non-cable environment, e.g., when no CableCARD is inserted.

In all cases, once an instance of an application starts running, its status as either an OCAP application or a non-OCAP application will be fixed and cannot change for the lifetime of that application instance.

When such applications are running as non-OCAP applications, the upper limit on the resources they can access will be the same resources as non-cable applications. The actual resources such an application instance can access MAY be more restricted than this. For example, if service bound applications run as non-OCAP applications then the rules and policies of the presenting environment determines which resources may be accessed by the application.

#### *10.2.2.5.2 Host Device Manufacturer Applications and the Cable Environment*

Non-cable applications SHALL execute concurrently with an OCAP-J proxy application and hence be host device manufacturer applications if either of the following apply;

- they access resources exclusively used by the cable environment (as defined in 19.2.1.2).
- they access shared resources (as defined in 19.2.1.2) when the cable environment is selected.

Non-cable applications to which neither of the above apply MAY be host device manufacturer applications but are not required to be.

OCAP implementations SHALL comply with the following in relation to host device manufacturer applications.

a. Services - The implementation SHALL install entries into the services list for each of the services associated with Host Device Manufacturer applications. Each entry contains the service_id that is used to associate the application with an abstract service. The entry also provides access to application attributes, for example whether or not the application is AUTO_START or PRESENT on selection of the abstract service. The information needed to provide this information is stored with the application in an implementation specific form. The lifecycle of these applications is not controlled by signaling.

b. Permissions - Where access to a function is controlled by the OCAP permission request file, the OCAP implementation SHALL allow a host device manufacturer application to perform only those functions that are permitted by the Java Permissions class associated with the OCAP-J proxy. Host device manufacturer applications written in Java that use the OCAP APIs but are not signaled using OCAP signaling are not required to be associated with a Permission Request File. However, such applications SHALL be associated with Java Permission classes. Host device manufacturer applications which access IP data via an IP network interface not directly connected to the cable network (e.g., an Ethernet or WiFi port) are not required to be associated with an instance of SocketPermission for this access.

c. Resource Access - The OCAP implementation SHALL allow a host device manufacturer application to access a shared resource subject to the DAVIC resource negotiation framework if the OCAP-J proxy has implemented the org.davic.resources.ResourceClient interface and uses the appropriate class implementing the org.davic.resources.ResourceProxy interface to reserve or otherwise interface with the resource. The OCAP implementation SHALL ensure that host device manufacturer applications honor any resource negotiation requests and resource release requests that they receive from the OCAP implementation, and SHALL successfully reserve the resource before use.

d. State Management - a host device manufacturer application SHALL only change the state of an shared resource or capability in a manner that is compatible with a similar change made from an OCAP-J application. When an OCAP-J application is monitoring events or state changes and a host device manufacturer application changes the state of the associated shared resource, the OCAP implementation SHALL provide the same indication of state change as when the change is made by an OCAP-J application.

e. Application Starting – Host device manufacturer applications with AUTO_START status SHALL be started when the service is selected. All changes to the lifecycle are managed either by the applications within the service or by the Monitor Application managing the application lifecycle through an application proxy or managing the service lifecycle.

f. Application Lifecycle - Host device manufacturer applications SHALL honor any lifecycle commands that are expressed using the AppProxy object that represents the OCAP-J Proxy.

#### *10.2.2.5.3 Graphics Interactions Between Manufacturer and OCAP applications*

OCAP implementations that permit simultaneous display of non-OCAP applications and OCAP applications SHALL comply with the following when this is happening:

a. The OCAP implementation SHALL ensure that only one application has input focus at one time.

b. The OCAP implementation SHALL ensure that when an OCAP application has focus, Annex K.2.2 is followed.

c. The OCAP implementation SHALL correctly generate java.awt.event.WindowEvent instances and repaints

when the state of the HScene of an OCAP application changes independent of whether this is due to a non-OCAP application or another OCAP application.

NOTE: There is no requirement for the top level UI container of non-cable applications to have an HScene. Top level UI containers without an HScene cannot by definition be managed by org.ocap.ui.HSceneManager.

#### *10.2.2.6 Application Priority*

A priority system is provided that can be used to provide an environment where all applications have a fair chance at accessing shared OCAP resources. Three types of applications are identified for the purposes of priority assignment; monitor, unbound, and service bound. These application types are implied by the priority they are assigned. The priority ranges used by OCAP 1.1 applications are described in Table 10–4.

***Table 10–4 - Priority Ranges for Application Types***

| Priority Range | Application Type |
|---|---|
| 255 | Initial Monitor Application |
| 100 to 254 | Unbound Application |
| 1 to 200 | Service Bound Application |

The application priority is a platform wide value and is used by the implementation to negotiate resources between various applications. One of these resources is the available memory to run the application and the implementation SHALL use the application priority to determine which applications will run when there is insufficient memory. Other resources are managed through the DAVIC resource management APIs and through the resource contention handler that the Monitor Application may have implemented. Resource management is discussed in Section 19, Resource Management.

The Monitor Application is the only application with priority 255, the highest available priority. This allows unrestricted access to all platform shared resources and the Monitor Application should take care to use these resources sparingly to avoid starving other applications.

Within the range of unbound application priorities, the highest priorities SHOULD be allocated to applications that take over the role of assumable system modules (see Section 20.2.4.11). This helps to ensure that these applications do not lose access to shared resources during a resource negotiation.

#### *10.2.2.7 Native API Access*

OCAP 1.1 applications MAY access functionality outside the OCAP API, known as native APIs. Native APIs MAY be part of an environment that is proprietary to the Host device. OCAP-J applications that access native APIs SHALL use the Java native interface (JNI) as specified in the Java Native Interface Specification ([JNI]), Release 1.1, 1996, Revised 1997 Part of ISBN:1-892488-25-6 [JVM Err], and JNI Enhancements in JDK1.2 Part of ISBN:1-892488-25-6 [JVM Err].

Native APIs that meet the criteria in Section 10.2.2.5.2, for a non-cable application to need an OCAP-J proxy SHALL be conditioned upon the calling OCAP-J application being granted access to those resources and SHALL perform in a manner that maintains the essential OCAP implementation state. Section 10.2.2.5 describes requirements for host device manufacturer applications that apply equally to native APIs.

#### *10.2.2.8 Content Handler Auto Launch*

The device manufacturer MAY supply applications that wish to automatically launch upon connection of removable storage or media with specific file types.

The selected environment determines what occurs upon the connection of removable media. If the content type on the removable media is not supported by OCAP, then the connection of the removable media MAY cause a switch to a CE environment. The CE environment is free to handle the content in any way it chooses.

See Section 10.2.2.4 for discussion of application environments and their selection.

Example:
OCAP does not have APIs for playing retail DVDs. If a retail DVD is inserted this MAY cause the selection of a CE environment. The CE environment MAY then begin playing the DVD.

#### *10.2.2.9 Broken or Unresponsive Applications*

In DVB-MHP 1.0.3 [9], section 9.1.4 defines a number of situations when applications either may or are required to be killed. OCAP 1.1 extends this as follows.

The implementation MAY monitor applications to detect those which are broken or otherwise have become unresponsive.

A complete list of criteria for this is not practical to define however example criteria include the following;

1. applications which attempt a denial of service attack on the host
2. applications exhibiting behavior resulting in a need to re-initialize the OCAP environment
3. applications which repeatedly call methods with illegal parameters
4. applications that block in event handler threads
5. applications that attempt to allocate all memory in the host
6. applications which block in an infinite loop
7. applications catching java.lang.ThreadDeath

*Editor's Note: All the above are place-holders for the purposes of discussion. Item #3 is already known to be particularly difficult.*

The implementation MAY decide to stop such applications. If such applications are re-launched and again exhibit the same broken or unresponsive behavior, the implementation MAY quarantine the application such that future attempts to run that application fail.

If a quarantined application is signaled with a different version number (e.g., signaled with an OCAP unbound application descriptor or an MHP 1.1 application storage descriptor), it SHALL automatically be removed from quarantine when an updated version is detected. Of course this new version may be put back in quarantine if it also exhibits broken or unresponsive behavior.

#### *10.2.2.10 Applications Modes*

Applications may exist in one of four modes – normal, cross-environment, background and paused. For applications which are Xlets, there is no relationship between the first three of these modes and the Xlet state machine. Xlets can be in any state in any of these three modes. An application in the paused mode which follows the Xlet state machine can only be in the paused state.

##### *10.2.2.10.1 Normal Mode*

An application is running in the normal mode when its home environment is either selected or presenting. Xlets in normal mode may be in any state – loaded, paused, active or destroyed.

#### *10.2.2.10.2Cross-environment Mode*

End-user interactions with cross-environment applications are typically transient, for example the acknowledgment of an event or the changing of some setting. Xlets in cross-environment mode may be in any state – loaded, paused, active or destroyed. Only unbound applications can run in cross-environment mode, bound applications shall not run in this mode.

Examples of manufacturer cross-environment applications operating when the cable is the selected environment include the following:

- utility applications such as volume control, settings, input selection
- reminders set via the manufacturer guide

Examples of cable cross-environment applications operating when manufacturer is the selected environment include the following;

- caller-id
- reminders set via the cable guide

By default, applications are not capable of running as cross-environment applications. Applications which can run as cross-environment must declare themselves as such via a mechanism described below. If the user-selects an application which can run as cross-environment and that application's environment is not the selected one, the application will start in as a cross-environment application. It can then decide whether to continue to run or to terminate or to offer the end-user the option of changing the selected environment to the application's home environment.

OCAP 1.1 host devices MAY provide applications in cross-environment mode with graphics resolutions other than those defined by this specification without the ability to change them. Under these circumstances, applications SHALL be given the correct HGraphicsConfiguration for the graphics resolution which will be used. Applications in cross-environment mode are responsible for checking this HGraphicsConfiguration if they wish to adapt their user interface for different graphics resolutions.

Applications which are visible when the selected environment changes and which are then put in to cross-environment mode are responsible for managing their presentation in the new environment. This includes visibility state and adjusting to changes in graphics resolution as described above.

#### *10.2.2.10.3Background Mode*

The following constraints apply to applications running in the background mode;

- they shall not have any graphics visible and any attempts to become visible shall fail (silently in the case of applications using the OCAP APIs)
- they shall not have focus and any attempts to request focus shall fail (silently in the case of applications using the OCAP APIs)
- they may request the reservation of shared resources, however whether a request succeeds will be decided by the policies of the selected environment, hence they must be prepared for these resource requests to fail and have a fallback strategy.
- they are not excluded from having DAVIC resources reserved which are shared resources, however whether they retain such resources will be decided by the policies of the selected environment, hence they must be prepared for these resources to be removed at any time and have a fallback strategy
- they SHOULD limit their use of CPU and memory resources.

An OCAP application running in the background mode can continue to use the embedded cable bi-directional channel since that is not a DAVIC resource. Both bound and un-bound applications can be paused.

#### *10.2.2.10.4Paused Mode*

Paused applications are OCAP applications which are signaled as pause-able and which have been put in the paused mode by the implementation when their home environment stops being selected. Pausing of applications is an implementation optimization to obtain a faster response when changing selected environment from cable to manufacturer and back. Both bound and un-bound applications can be paused.

Paused applications differ from applications running in background mode in the following ways;

- they are not providing any useful function. Applications in background mode may be providing a useful function such as gathering guide data or listening for incoming messages such as IM or caller-id.
- Their Xlet state has been changed to paused and the pauseXlet method called.
- They may be starved of processor cycles, e.g., by assigning a very low priority to any threads running code from that application

Signaling an application as pause-able means that application has been tested for repeated transitions from the Active state to the Paused state and back to the Active state. Typically this will mean an implementation of the pauseXlet method which is not empty and an implementation of the startXlet method which can handle being called both when the Xlet is first starting and subsequently when the Xlet is coming out of the Paused state.

When designating which applications are background or cross-environment, the application provider is responsible for ensuring that these will fit within the defined available resources for OCAP in the background state. In case they do not fit, the implementation MAY terminate OCAP applications in ascending order of priority until the remaining set of applications will fit within the defined resources.

# 11 APPLICATION SIGNALING

This section contains requirements that are extensions to [DVB-GEM 1.1] Section: 10 Application signaling and are extensions of [DVB-MHP1.1.2] Section: 10 Application Signaling.

This section covers how the OCAP 1.1 terminal identifies OCAP 1.1 applications associated with a service and how the terminal finds the locations from which to retrieve them. This section also identifies the signaling that enables the broadcast to manage the life cycles of bound and unbound applications. Finally, this section shows how the receiver can identify the sources of broadcast data required by the bound applications of a service.

## 11.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section. Section 11 (this section) of the OCAP 1.1 Profile corresponds to Section 10 of [DVB-MHP1.1.2] and [DVB-GEM 1.1] as detailed in Table 11–1.

When reading [DVB-MHP1.1.2] or [DVB-GEM 1.1] with regard to OCAP 1.1 correspondence, the term "DVB" or "GEM" SHALL be read as "OCAP 1.1", and the term "DVB-J" SHALL be read as "OCAP-J"

***Table 11–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 11 Application Signaling | 10 Application signaling | Extension | 10 Application signaling | Extension |
| 11.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 11.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 11.2.1 Deviations from the DVB-MHP Specification | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 11.2.1.1 Introduction | 10.1 Introduction | Extension | 10.1 Introduction | Extension |
| No Corresponding Section | 10.2 Program specific information | Compliance | 10.2 Program specific information | A subsection is extended |
| No Corresponding Section | 10.2 Program specific information | Compliance | 10.2.1 Application signaling stream | Compliance |
| 11.2.1.2 Data broadcast streams | 10.2 Program specific information | Compliance | 10.2.2 Data broadcast streams | Extension |
| No Corresponding Section | 10.3 Locators within an application description | Compliance | No Corresponding Section | |
| No Corresponding Section | 10 Application signaling | Compliance | 10.3 Notation | Compliance |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| No Corresponding Section | 10.4 Application description | A subsection is extended | 10.4 Application Information Table | A subsection is extended |
| No Corresponding Section | 10.4 Application description | Compliance | 10.4.1 Data Errors | Compliance |
| No Corresponding Section | 10.4.1 Application Description Transmission and Monitoring | Compliance | 10.4.2 AIT transmission and monitoring | Compliance |
| No Corresponding Section | 10.4 Application description | Compliance | 10.4.3 Optimized AIT signaling | Compliance |
| No Corresponding Section | 10.4.2 Visibility of application description | Compliance | 10.4.4 Visibility of AIT | Compliance |
| No Corresponding Section | 10.4.3 Content of the application description | Compliance | No Corresponding Section | |
| No Corresponding Section | 10.4 Application description | Compliance | 10.4.5 Definition of sub-table for the AIT | Compliance |
| 11.2.1.3 Syntax of the AIT | 10.4 Application description | Compliance | 10.4.6 Syntax of the AIT | Extension |
| No Corresponding Section | 10.4 Application description | Compliance | 10.4.7 Use of private descriptors in the AIT | Compliance |
| No Corresponding Section | 10.4 Application description | Compliance | 10.4.8 Text encoding in AIT | Compliance |
| No Corresponding Section | 10. Application Signaling | Compliance | 10.5 Application identification | A subsection is extended |
| No Corresponding Section | 10. Application Signaling | Compliance | 10.5.3 Authentication of application identification | Compliance |
| No Corresponding Section | 10. Application Signaling | Compliance | 10.6 Control of application life cycle | Compliance |
| No Corresponding Section | 10. Application Signaling | Compliance | 10.7 Generic descriptors | Compliance |
| | 10. Application Signaling | Compliance | 10.8 Transport protocol descriptors | Compliance |
| 11.2.1.4Transport Protocol Descriptor | 10. Application Signaling | Compliance | 10.8.1 Transport protocol descriptor | Extension |
| 11.2.1.5 Transport via OC | 10. Application Signaling | Compliance | 10.8.1.1 Transport via OC | Extension |
| 11.2.1.6 Transport via IP | 10. Application Signaling | Compliance | 10.8.1.2 Transport via IP | Extension |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 11.2.1.7 Transport via Interaction Channel | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | OCAP-Specific Extension |
| No Corresponding Section | 10. Application Signaling | Compliance | 10.8.3 Pre-fetch signaling | Compliance |
| No Corresponding Section | 10.5 DVB-J specific application description | Compliance | 10.9 DVB-J specific descriptors | Compliance |
| No Corresponding Section | 10. Application Signaling | Compliance | 10.10 DVB-HTML Specific descriptors | Compliance |
| 11.2.1.8 Constant values | 10. Application Signaling | Compliance | 10.11 Constant values | Extension |
| 11.2.1.9 Service Information | 10. Application Signaling | Compliance | 10.12 Service Information | Extension |
| 11.2.2 Extensions to DVB-GEM (Normative) | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| No Corresponding Section | 10.6 Constant Values | Compliance | No Corresponding Section | |
| No Corresponding Section | 10.7 Plug-in Signaling | Compliance | 10.13 Plug-in signaling | Compliance |

## 11.2 OCAP 1.1 Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

### 11.2.1 Deviations from the DVB-MHP Specification

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

#### *11.2.1.1 Introduction*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 10.1 Introduction and are extensions of [DVB-MHP1.1.2] Section: 10.1 Introduction.

Signaling is used in OCAP 1.1, like DVB-MHP, to affect the lifecycle of service bound applications. In addition to DVB-MHP AIT signaling, this chapter has been extended beyond Section 10 of the [DVB-MHP1.1.2] and [DVB-GEM 1.1] to cover signaling of unbound applications.

OCAP 1.1 supports multiple forms of signaling to support service bound and unbound applications. The result of this signaling creates and/or updates entries in the Application Database. The Application Database is a collective name for the data kept by the application manager to support the Application Lifecycle APIs (refer to Annex N).

There are three forms of application signaling in OCAP 1.1; one for service bound applications; the other two are for unbound applications.

Service bound application signaling includes:

- DVB-MHP AIT signaling in the PSI of an in-band transport

Unbound application signaling includes:

- XAIT signaled in an Extended Channel MPEG section flow, defined in [CCIF 2.0], Section 9.14.
- OCAP 1.1 registrar APIs enabling the Monitor Application to create unbound applications

The three forms of signaling defined above use different encoding of essentially the same abstract data structure, the Application Information Table (AIT). The encoding of the AIT is as follows:

- DVB-MHP uses a binary encoding of the AIT as an extension of the PMT in the PSI. The service to which the AIT belongs is implicitly the service under which the application SHALL run.
- At boot time, the application manager needs to know which application(s) to start, in particular which Monitor Application(s). This information is carried in an OOB XAIT.
- The Monitor Application(s) uses the `org.ocap.application` APIs to manage the lifecycle of unbound applications.

Signaling for service bound applications adheres to the [DVB-MHP1.1.2]. Signaling for OCAP 1.1 unbound applications adheres to the OCAP 1.1 Profile as defined in this section. The Monitor Application uses the `org.ocap.application` APIs to manage the lifecycle of unbound applications.

#### *11.2.1.2 Data broadcast streams*

This subsection is compliant with [DVB-GEM 1.1] Section: 10.2 Program specific information and are extensions of [DVB-MHP1.1.2] Section: 10.2.2 Data broadcast streams.

Minimum signaling in the PMT is defined by [SCTE 40], not [EN 301 192].

#### *11.2.1.3 Syntax of the AIT*

This subsection is compliant with [DVB-GEM 1.1] Section: 10.4 Application description and are extensions of [DVB-MHP1.1.2] Section: 10.4.6 Syntax of the AIT.

The OCAP 1.1 Profile complies with Section 10.4.6 of the [DVB-MHP1.1.2] except where stated below.

Application types SHALL be modified so that application_type 0x0001 is described as an OCAP-J application. Application_type 0x0002 DVB-HTML is out of scope of OCAP 1.1. Applications with application_type in the range 0x0002 to 0xFFFF are not supported by this standard and may be ignored.

#### *11.2.1.4 Transport Protocol Descriptor*

This subsection is compliant with [DVB-GEM 1.1] Section: 10. Application Signaling and are extensions of [DVB-MHP1.1.2] Section: 10.8.1 Transport protocol descriptor.

OCAP 1.1 differs from Section 10.8.1 of the [DVB-MHP1.1.2] in the values defined for the protocol_id.

OCAP 1.1 defines the protocol_id value of 0x0101 as IP via two-way interactive channel.

Protocol_id of [DVB-MHP1.1.2] is replaced with the following Table to modify reference sections:

***Table 11–2 - Protocol_id***

| protocol_id | Description |
|---|---|
| 0x0000 | reserved_future_use |
| 0x0001 | OCAP Object Carousel as defined in OCAP 1.1, Section 22 |
| 0x0002 | IP via DVB multi protocol encapsulation. Not supported by OCAP 1.1 |
| 0x0003 | Reserved for MHP1.1 |
| 0x0004...0x00FF | Reserved_future_use |
| 0x0100 | Reserved |
| 0x0101 | See Section 11.2.2.3.9 Transport via Interaction Channel |
| 0x0102...0xFFFF | Subject to registration in TR 101 162 |

Semantic of selector bytes of [DVB-MHP1.1.2] is replaced with the following Table to modify reference sections:

***Table 11–3 - Semantic of selector bytes***

| protocol_id | Description |
|---|---|
| 0x0000 | reserved_future_use |
| 0x0001 | See OCAP 1.1, Section 11.2.1.5 Transport via OC for an AIT case and Section 11.2.2.3.7 Transport via OC for an XAIT case. |
| 0x0002 | See OCAP 1.1, Section 11.2.2.3.8 Transport via IP for an XAIT case. Not supported. |
| 0x0003...0x0100 | Not defined in this version of the specification |
| 0x0101 | See OCAP 1.1, Section 11.2.2.3.9 Transport via Interaction Channel for an XAIT case. |
| 0x0102...0xFFFF | Not defined in this version of the specification |

**Note:** If an application has been stored in application storage according to Section 12, the transport_protocol_descriptor SHALL be ignored when launching the application and the application files SHALL be read from application storage. In this case, ServiceDomain SHALL NOT be attached even if the protocol_id = 0x0001 (Object Carousel) and the application files SHALL NOT be updated unless explicit application signaling in the XAIT so specifies.

### *11.2.1.5 Transport via OC*

This subsection is compliant with [DVB-GEM 1.1] Section: 10. Application Signaling and are extensions of [DVB-MHP1.1.2] Section: 10.8.1.1 Transport via OC.

OCAP 1.1 differs from Section 10.8.1.1 of [DVB-MHP1.1.2] in the definition of the selector bytes for the OC transport.

If the remote_connection is 1, the original_network_id and transport_stream_id default to zero and are ignored. The service_id value is used as the source_id.

The selector_bytes of Transport via OC indicates only an in-band DSMCC Object Carousel. OOB Object Carousel is out of scope regarding to OCAP-J application transmission for AIT signaled bound applications.

#### *11.2.1.6 Transport via IP*

This subsection is compliant with [DVB-GEM 1.1] Section: 10. Application Signaling and are extensions of [DVB-MHP1.1.2] Section: 10.8.1.2 Transport via IP.

This section corresponds to Section 10.8.1.2 of [DVB-MHP1.1.2], OCAP 1.1 doesn't support DVB multi- protocol encapsulation.

#### *11.2.1.7 Transport via Interaction Channel*

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] section.

The selector bytes for transport protocol_id = 0x0101 in the transport protocol descriptor shall be as specified in Table 11–4.

***Table 11–4 - Syntax of selector bytes for interaction channel***

| Syntax | Bits | Mnemonic |
|---|---|---|
| URL_length<br>for( i=0; i< URL_length; i++) {<br>URL_byte<br>}<br>} | 8<br><br>8 | uimsbf<br><br>uimsbf |

URL_length: This 8-bit field provides the number of bytes in the base part of the URL.

URL_byte: These bytes form the HTTP URL to a top-level directory to be mounted. It SHALL be a UTF8 string without null termination. Only HTTP 1.1 protocol on TCP/IP is supported according to Section 8. For example, URL_byte represents a string of "http://192.168.0.123/app/". Files under the top-level directory are downloaded individually by HTTP protocol (i.e., each class file is fetched by a separate HTTP "GET" request). For example, file access via a java.io.File causes an HTTP access to the files under the specified URL in the specified remote server.

In case that a signed application is stored in application storage, all files in an application description file SHALL be authenticated prior to launching. (See also Section 12.2.7.) Authentication of files on a remote HTTP server is not allowed. Wild card ("*") cannot be specified in the application description file since HTTP protocol cannot get a list of files in a directory and cannot distinguish a file name from a directory name.

In case that a signed application is not stored in application storage, all files described in all hash files of the application SHALL be downloaded via HTTP protocol, cached in the host, and authenticated prior to launching. Authentication of files on a remote HTTP server is not allowed.

Applications downloaded from such an IP URL SHALL be listed in the applications database and be launched in the same fashion as any other application so listed.

##### *11.2.1.7.1 HTTP Redirection*

During application download, a server may respond to an HTTP GET request with a redirection response where the HTTP status code is in the 3xx range. The following behaviors SHALL be adhered to by an HTTP user agent that is part of an implementation when a redirection status is received in response to an HTTP GET request during application download:

- When a redirection status code in the 3xx range is received for a GET request and an address for the redirected server is included in the response, the implementation SHALL respond to the response by

resending the GET request using the server address indicated in the redirection response. This is also known as an automatic redirect.

- When status code 300 or 301 is received by the implementation and the message contains multiple server addresses, the implementation SHALL choose the server address in an implementation-specific fashion.

The user agent SHOULD detect for infinite redirections.

#### *11.2.1.8 Constant values*

This subsection is compliant with [DVB-GEM 1.1] Section: 10. Application Signaling and are extensions of [DVB-MHP1.1.2] Section: 10.11 Constant values.

OCAP 1.1 differs from Section 10.11 of [DVB-MHP1.1.2] as follows:

The DVB-HTML descriptors are out of scope in OCAP 1.1 and SHALL NOT be used.

#### *11.2.1.9 Service Information*

This subsection is compliant with [DVB-GEM 1.1] Section: 10. Application Signaling and are extensions of [DVB-MHP1.1.2] Section: 10.12 Service Information. See Section 8.2.1.4 for further information.

### 11.2.2 Extensions to DVB-GEM (Normative)

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

The following information is normative for the OCAP 1.1 Profile. It is specific to OCAP 1.1 and is not contained within the [DVB-MHP1.1.2].

#### *11.2.2.1 Signaling of Unbound Applications*

OCAP 1.1 application signaling of MSO applications is used to announce and control MSO applications.

##### *11.2.2.1.1 Control of MSO Application Lifecycle*

Refer to Section 10.2.2.3 for a description of how signaling effects the lifecycle of unbound applications.

##### *11.2.2.1.2 Delivery of Signals for MSO Applications*

MSO applications, not being associated with a service that provides AIT PSI in-band signaling, require a different signaling method.

AIT type information is provided to MSO applications by either OCAP 1.1 application signaling via the OCAP 1.1 XAIT or via utilization of AppManagerProxy.registerUnboundApp APIs. The XAIT contains many of the application related fields that are found in the in-band AIT as well as additional information to control MSO Application behavior (for example, application storage). The format of the XAIT is described in Section 12.

#### *11.2.2.2 Unbound Applications and the Application Database*

The Application Database contains information for the currently available broadcast applications and the unbound applications associated with selected abstract services. Refer to Section 10.2.2.2.2 and Section 21.2.1.8 for details.

#### *11.2.2.3 OCAP 1.1 XAIT*

The XAIT signals unbound applications (i.e., applications that are not bound to an in-band service). The XAIT is extended from the AIT defined in chapter 10 of [DVB-MHP1.1.2]. The maximum cycle time of the XAIT SHOULD be 10 seconds. Implementation SHALL continuously monitor for presence of the XAIT and SHALL immediately detect every XAIT change. When the cable environment is in the selected or presenting state, the implementation SHALL immediately detect a version change in the XAIT when an XAIT is present; when the cable environment is in the background state the implementation SHALL detect an XAIT update at least once every 30 seconds when an XAIT is present; when the cable environment is in the inactive state the implementation SHALL detect an XAIT update at least once every 1 hour. The implementation must listen for XAIT updates even when the cable environment is inactive to accommodate the case where a network previously has not signaled an XAIT but then subsequently does, as when OCAP services are newly introduced into a region. When transmitted, XAIT Table sections will be placed in an Extended Channel MPEG section flow using PID 0x1FFC with the same table_id as the in-band AIT. The XAIT table sections SHALL have the format specified by Section 10.4.6 of the [DVB-MHP1.1.2] of the AIT, with the constraints as specified by Sections 11.2.1.3 and 11.2.1.5. Applications signaled in an XAIT SHOULD NOT be signaled as REMOTE in the application's application control_code field. The implementation SHALL interpret an XAIT application_control_code value of REMOTE as PRESENT. Entries in any AppsDatabase SHALL be created with the value of PRESENT.

Even though the XAIT has the same Table Id as the AIT it is recognized as a XAIT based on its location in an Extended Channel MPEG section flow.

The abstract service descriptor needs to be repeated in each sub-Table of the XAIT (e.g., if there is more than one application type then the XAITs for each application type SHALL have their own abstract service descriptor(s)). The set of abstract services introduced in each XAIT sub-Table is not required to be the same. However, the allocation of service identifiers to abstract services SHALL match when different sub-Tables reference the same abstract service.

##### *11.2.2.3.1 Application_Id Values*

The application_id values associated with unbound applications are divided into three ranges: one for unsigned applications, one for signed applications, and one for dual signed applications. Applications transmitted as unsigned shall use an application_id from the unsigned applications range, applications not requiring monitor privileges and transmitted as signed shall use an application_id from the signed applications range, applications requiring monitor privilege shall use an application_id from the dual signed applications range. Application_id values 0xffff and 0xfffe are reserved and are not used by unbound applications.

***Table 11–5 - Value ranges for application_id in unbound applications***

| application_id values | Use |
|---|---|
| 0x0000 0x3fff | Application_ids for unsigned applications |
| 0x4000 0x5fff | Application_ids for signed applications without monitor permission |
| 0x6000 0x7fff | Application_ids for dual signed applications that use monitor permission |
| 0x8000 0xfffd | Reserved for future use by DVB |
| 0xfffe 0xffff | Reserved |

#### *11.2.2.3.2 XAIT Descriptors*

The XAIT constrains and extends the descriptor usage definition of the AIT. Unless otherwise indicated, descriptors contained within a XAIT SHALL be contained within the inner application descriptor loop. XAITs which contain errors will be processed according to the rules specified in Section 10.4.1 of the [DVB-MHP1.1.2] Data Errors.

#### *11.2.2.3.3 Application Descriptor*

An application descriptor as specified in Section 10.7.3 of the [DVB-MHP1.1.2] SHALL be included in the XAIT for each application signaled. OCAP 1.1 extends the definition of the application descriptor as described below:

> An unbound application can only be associated with a single abstract service. Thus, the service_bound_flag SHALL always be set to 1.

#### *11.2.2.3.4 Application Name Descriptor*

The application name descriptor SHALL be included in the XAIT in the same manner specified for the AIT in Section 10.7.4.1 of the [DVB-MHP1.1.2].

#### *11.2.2.3.5 Application Icons Descriptor*

The application icons descriptor may be included in the XAIT in the same manner specified for the AIT in Section 10.8 of the [DVB-MHP1.1.2].

#### *11.2.2.3.6 Transport Protocol Descriptor*

The transport protocol descriptor SHALL be included in the XAIT in the same manner specified for the AIT in Section 10.8 of the [DVB-MHP1.1.2], and, as deviated by in OCAP 1.1 Section 11.2.1.4.

#### *11.2.2.3.7 Transport via OC*

When the protocol_id of the transport protocol descriptor is 0x0001 the selector bytes SHALL be included in the XAIT in the same manner specified for the AIT in Section 10.8.1.1 of the [DVB-MHP1.1.2], and as deviated by in OCAP 1.1 Section 11.2.1.6. In addition, when used in a XAIT, the remote_connection field SHALL be 1 whether the XAIT is delivered via the Extended Channel or specified by an AppManagerProxy.registerUnboundApp() method regardless of current selected service signaling, as defined in Section 22.2.2.

Applications downloaded from such an OC SHALL be listed in the applications database and SHALL be launched in the same manner as any other application so listed.

#### *11.2.2.3.8 Transport via IP*

OCAP 1.1 doesn't support the protocol_id 0x0002 of the transport protocol descriptor.

#### *11.2.2.3.9 Transport via Interaction Channel*

When the protocol_id of the transport protocol descriptor is 0x0101 the selector bytes SHALL be included in the XAIT in the same manner specified for the AIT in OCAP 1.1 Section 11.2.1.7.

#### *11.2.2.3.10Pre-fetch Descriptor*

Pre-fetch descriptors may be included in the XAIT in the same manner specified for the AIT in Section 10.8.3.2 of the [DVB-MHP1.1.2].

#### *11.2.2.3.11DII location Descriptor*

DII location descriptors may be included in the XAIT in the same manner specified for the AIT in Section 10.8.3.3 of the [DVB-MHP1.1.2].

#### *11.2.2.3.12OCAP-J Application Descriptor*

The OCAP-J Application descriptor SHALL be included in the XAIT in the same manner specified for the AIT in Section 10.9.1 of the [DVB-MHP1.1.2], and as deviated by in OCAP 1.1 Section 11.2.1.1, all occurrences of DVB-J SHALL be recognized as an OCAP-J application type in an OCAP 1.1 environment.

#### *11.2.2.3.13OCAP-J Application Location Descriptor*

The OCAP-J Application descriptor SHALL be included in the XAIT in the same manner specified for the AIT in Section 10.9.2 of the [DVB-MHP1.1.2], and as deviated by in OCAP 1.1 Section 11.2.1.1, all occurrences of DVB-J SHALL be recognized as an OCAP-J application type in an OCAP 1.1 environment.

#### *11.2.2.3.14Abstract Service Descriptor*

OCAP 1.1 extends [DVB-MHP1.1.2] and defines the abstract service descriptor. One abstract service descriptor is required for each abstract service that may be selected from the network. One or more abstract service descriptors SHALL be contained within the common descriptor loop of the XAIT.

***Table 11–6 - Abstract Service Descriptor***

| | No. of Bits | Identifier | Value |
|---|---|---|---|
| abstract_service_descriptor() { | | | |
| descriptor_tag | 8 | uimsbf | 0x66 |
| descriptor_length | 8 | uimsbf | |
| service_id | 24 | uimsbf | |
| reserved_for_future_use | 7 | uimsbf | |
| auto_select | 1 | bslbf | |
| for (i=0; i<N; i++) { | | | |
| service_name_byte | 8 | uimsbf | |
| } | | | |
| } | | | |

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

descriptor_length: Identifies the number of bytes immediately following the length field.

service_id: Service identifier for the abstract service. This is 24 bit value to avoid conflict with the 16 bit service_id for a broadcast service. The following value range shall be applied.

***Table 11–7 - Service_id Value Range***

| | service_id [24 bit] |
|---|---|
| Abstract service defined by MSO | 0x020000 - 0xFFFFFF (24 bit) |
| Abstract service defined by manufacturer | 0x010000 - 0x01FFFF (24 bit) |

**auto_select:** If set to 1, indicates the service SHALL be automatically selected. See Section 10.2.2.2.2.2 for complete auto-select definition.

**service_name_byte:** One UTF-8 character in the abstract service name. Taken together, all of the service_name_byte fields make up the abstract service name. The abstract service name is not null terminated.

#### *11.2.2.3.15Unbound Application Descriptor*

OCAP 1.1 extends [DVB-MHP1.1.2] and defines the unbound application descriptor. Exactly one unbound application descriptor SHALL be contained in the application descriptor loop for each application signaled in the XAIT.

***Table 11–8 - Unbound Application Descriptor***

| | No. of Bits | Identifier | Value |
|---|---|---|---|
| unbound_application_descriptor() { | | | |
| descriptor_tag | 8 | uimsbf | 0x67 |
| descriptor_length | 8 | uimsbf | |
| service_id | 24 | uimsbf | |
| version_number | 32 | uimsbf | |
| } | | | |

**descriptor_tag:** This 8 bit integer with value 0x67 identifies this descriptor.

**descriptor_length:** Identifies the number of bytes immediately following the length field.

**service_id:** Service ID of the abstract service this application belongs to. The service_id SHALL match a service defined in one of the abstract service descriptors.

**version_number:** Version number of this application. Once the version number reaches the maximum value and an increment is needed, a new application_id SHALL be assigned to the application.

#### *11.2.2.3.16Privileged Certificate Descriptor*

OCAP 1.1 extends [DVB-MHP1.1.2] and defines the privileged certificate descriptor. Exactly one privileged certificate descriptor SHALL be contained in the common loop in the XAIT.

***Table 11–9 - Privileged Certificate Descriptor***

| | No. of Bits | Identifier | Value |
|---|---|---|---|
| privileged_certificate_descriptor () { | | | |
| descriptor_tag | 8 | uimsbf | 0x68 |
| descriptor_length | 8 | uimsbf | |
| for (i=0; i<N; i++) { | | | |
| for (j=0; j<20; j++) { | | | |
| certificate_identifier_byte | 8 | uimsbf | SHA-1 Hash |
| } | | | |
| } | | | |

**descriptor_tag:** This 8 bit integer with value 0x68 identifies this descriptor.

**descriptor_length**: Identifies the number of bytes immediately following the length field.

**certificate_identifier:** A SHA-1 hash of a certificate (in DER encoded form) that is used to sign an application that needs monitor permission. One or more such SHA-1 hashes may be included to identify authorized certificates.

### *11.2.2.3.17Application Storage Descriptor*

OCAP 1.1 extends [DVB-MHP1.1.2] and defines the application storage descriptor. For each application to be stored in persistent storage, one application storage descriptor SHALL be contained in the application descriptor loop in the XAIT. If an application storage descriptor is not present for an application, a storage priority of 0 (never store), will be assigned to that application.

***Table 11–10 - Application Storage Descriptor***

| | No. of Bits | Identifier | Value |
|---|---|---|---|
| application_storage_descriptor() { | | | |
| descriptor_tag | 8 | uimsbf | 0x69 |
| descriptor_length | 8 | uimsbf | |
| storage_priority | 16 | uimsbf | |
| launch_order | 8 | uimsbf | |
| } | | | |

**descriptor_tag:** This 8 bit integer with value 0x69 identifies this descriptor.

**descriptor_length:** Identifies the number of bytes immediately following the length field.

**storage_priority**: Storage priority of the application, see Section 12.2.3.1.

**launch_order:** Order of applications with the same application identification and priority. Only the application with the highest launch order is entered in the Application Database. The launchOrder may be used to signal and store a new version of an application prior to a change of launchOrder in a subsequent revision of the XAIT.

#### *11.2.2.3.18Registered API Descriptor*

The ocap_j_registered_api_descriptor requests application access to the shared classes of a registered API. This descriptor can be contained in the application descriptors loop of an AIT or XAIT. Up to 16 of these descriptors MAY be carried in an application's descriptor loop. If more than 16 instances of this descriptor are present in a single application's descriptor loop, then the terminal MAY ignore all instances of this descriptor after the 16th one. Section 21.2.1.20 specifies usage rules for this descriptor.

***Table 11–11 - Registered API Descriptor***

| | No. of Bits | Identifier | Value |
|---|---|---|---|
| ocap_j_registered_api_descriptor() { | | | |
| descriptor_tag | 8 | uimsbf | 0x6A |
| descriptor_length | 8 | uimsbf | |
| for( i=0; i<N; i++ } { | | | |
| registered_api_name_char | 8 | uimsbf | |
| } | | | |
| } | | | |

**descriptor_tag:** This 8 bit integer with value 0x6A identifies this descriptor.

**descriptor_length:** Number of bytes immediately following the length field.

**registered_api_name_char:** One character in a registered API name. Taken as a whole, these characters represent the registered API name that the application is requesting access to. This string is encoded using Java's modified UTF-8 encoding and SHALL NOT be terminated with a zero byte. Uniqueness of registered API names SHOULD be managed by prefixing them with the internet domain name of the organization providing the registered API.

#### *11.2.2.3.19Application mode descriptor*

OCAP 1.1 extends DVB-MHP 1.0.3 [9] and defines the application mode descriptor. Zero or one application mode descriptors SHALL be contained in the application descriptor loop for each application signaled in the XAIT or in the AIT.

***Table 11–12 - Application Mode Descriptor***

| | No. of bits | Identifier | Value |
|---|---|---|---|
| application_mode_descriptor() { | | | |
| descriptor_tag | 8 | uimsbf | 0x70 |
| descriptor_length | 8 | uimsbf | |
| Mode | 2 | uimsbf | |
| Reserved | 6 | uimsbf | |
| } | | | |

The values of the mode variable are as follows:

0 Indicates that the application can only run in normal mode.

1 Indicates that the application can run in cross_environment mode. This flag SHALL be ignored for applications signaled in an AIT (i.e., applications not signaled in an XAIT).

2 Indicates that the application can run in background mode.

3 Indicates that the application can run in paused mode.

# 12 APPLICATION STORAGE

This section contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

This section describes the management of applications that are stored on the OCAP 1.1 terminal. This covers the protocols used in the storage of applications from different storage and the manner in which the terminal manages the replacement and removal of applications as new versions are issued.

## 12.1 DVB-GEM and DVB-MHP Specification Correspondence

This section of the OCAP 1.1 Profile does not correspond to any section in the [DVB-MHP1.1.2] or [DVB-GEM 1.1].

***Table 12–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 12 Application Storage | 10.8 Stored Applications | Extension | 10.14 Stored applications | Extension |

## 12.2 OCAP 1.1 Profile Requirements

### 12.2.1 Introduction

The loading and launching of an OCAP 1.1 application may benefit from the application being stored on the terminal. An OCAP 1.1 terminal SHALL provide the capability to store applications and to manage the storage assigned to applications. The size of the storage is not defined by OCAP 1.1 and is implementation dependent. OCAP applications SHALL NOT be able to write to the storage used for applications. The file hierarchy used for persistent storage of applications SHALL NOT be below the directory defined by the "dvb.persistent.root" property.

The applications that can be stored on an OCAP 1.1 terminal are sourced from the following set of different sources:

This section defines the rules for storage management for applications provided from each of these sources except the broadcast service. OCAP 1.1 does not support the storage of applications delivered via a broadcast service.

### 12.2.2 Storing Terminal Manufacturer Applications

The terminal manufacturer can install applications at the point of manufacture or at the time of download of a new implementation image or partial image through [CCIF 2.0]. Each of these applications will be identified by a common organization_id and a unique application_id. When a new implementation image is installed, all previously stored applications for the specified organization_id are removed and the new set of applications is installed. When a partial image is installed, those applications that are listed to be replaced or removed are removed from storage and the set of applications provided in the partial image is installed. The terminal manufacturer SHALL ensure that there is sufficient storage capacity to allow for the installation of the application set delivered in the download image.

The terminal manufacturer SHALL ensure that applications with its organization_id are not removed except as part of the implementation image installation process.

### 12.2.3 MSO applications signaled in the XAIT

#### *12.2.3.1 Storing applications*

The MSO signals a request to store an application by using the storage_priority field in the XAIT signaling. The storage_priority can take the following values:

***Table 12–2 - Storage Applications***

| storage_priority value | Meaning |
|---|---|
| 0 (default) | This application SHALL NOT be stored |
| 1-10 | Reserved for future use |
| 11-255 | This application is stored according to the specified priority value and subject to available storage constraints. |

The implementation SHALL store applications in order of storage_priority (highest value first) and remove unlaunched applications of lower storage priority to increase storage capacity when needed for higher storage priority applications. When there is insufficient storage capacity to store all applications signaled with the same storage priority, it is implementation specific which of these applications are stored. It is implementation specific whether currently running applications of lower storage priority are removed to release space for applications with a higher storage priority. In any case, the storing of applications SHALL NOT adversely affect the behavior of applications that are currently running.

When the XAIT is modified and an application is signaled with a higher storage priority than in the previous version of the XAIT, the implementation SHALL re-evaluate the need to store the application according to this revised storage priority. For example, the original storage priority may have been too low to enable the application to be stored given the state of the receiver. If the new storage priority enables the application to now be stored, it SHALL be stored subject to the constraints in the previous paragraph.

#### *12.2.3.2 Removing applications*

Applications SHALL be removed from storage in the following circumstances:

a. There is insufficient storage capacity for an application signaled with a higher storage priority (see previous subsection)

b. The application storage priority is set to zero or is omitted from the XAIT

c. The application is no longer signaled in the XAIT

An application whose storage priority is reduced to a non-zero value in a modified version of the XAIT is not removed from storage unless there is insufficient capacity for applications with a higher storage priority.

The implementation MAY remove applications from storage in other implementation defined circumstances.

#### *12.2.3.3 Managing version changes in applications*

When an MSO wishes to replace a currently stored application with a new version, it SHALL signal the same application_id with a new application version in the XAIT.

The MSO MAY continue to signal an old version of the application in the XAIT with a higher launch order than the new version. This allows an MSO to download a new version of the application before signaling that the new version is the one to be launched.

If the old version of the application is not currently running and is either not signaled or signaled with storage priority zero in the XAIT, the terminal will remove the old version and the new version is installed according to its storage priority.

If the old version of the application is currently running and is either not signaled or signaled with storage priority zero in the XAIT, it is implementation dependent whether this old version is removed before the application terminates. In this case both the old version and the new version of the application MAY be concurrently stored until the old version terminates. If the implementation removes the old version of a currently launched application, the continued behavior of the application SHALL NOT change during its current life cycle. A change in application version is indicated as an APP_CHANGED event to any registered listeners for AppsDatabaseEvents. The listening application can call the OcapAppAttributes.hasNewVersion() method to find whether the indicated event is caused by a new version of the application being registered in the Application Database. The recipient of such an event MAY decide to inform the user that a new version of the application is available and will be activated once the application terminates and is re-launched.

### 12.2.4 MSO applications installed through the Monitor Application

#### *12.2.4.1 Storing applications*

The Monitor Application determines the storage priority that should be associated with an application that it registers in the Application Database. This storage priority is made available as an attribute in the XAIT fragment used in the AppManagerProxy.registerUnboundApp(xait) method. The implementation uses this storage priority to manage the storage of this application using the same rules as those specified for applications signaled in the XAIT. Application storage is initiated as part of the registration process.

The storage priority is a common value for all applications irrespective of whether they are signaled in the XAIT or registered by the Monitor Application.

#### *12.2.4.2 Removing applications*

Applications SHALL be removed from storage in any of the following circumstances:

a. There is insufficient storage capacity for an application signaled with a higher storage priority (see previous subsection)

b. The application storage priority is set to zero in a new call to AppManagerProxy.registerUnboundApp()

c. The application that registered this application calls AppManagerProxy.unregisterUnboundApp() for this application

An application whose storage priority is reduced to a non-zero value in a call to AppManagerProxy.registerUnboundApp() is not removed from storage unless there is insufficient capacity for applications with a higher storage priority.

The implementation MAY remove applications from storage in other implementation-defined circumstances. For example, in the case that the application that registered a stored application is destroyed. This would allow the implementation to control storage of applications that have been registered and neither unregistered nor used since they were initially stored.

Note that an implementation SHOULD retain stored applications across reboots of the terminal. The period for which such stored applications are retained is implementation-defined and may depend on the set of applications signaled or registered once the terminal resumes operation.

### 12.2.5 Populating the SI Database

During the boot process the implementation SHALL populate the SI Database with the following information:

a. For each receiver manufacturer application, an entry is included in the database and the control code for the application is set from information stored with the application by the manufacturer. The format of this stored information is implementation specific, but could be in the form of a XAIT fragment.

b. For each MSO application signaled in the XAIT, an entry containing the signaling information is included in the database.

c. For each MSO application registered by the Monitor Application, an entry containing the signaling information stored at the time of registration is included in the database.

All of these entries are included in the database prior to the launch of the Monitor Application. This allows the Monitor Application access to detailed information for all available stored applications.

### 12.2.6 Removal of Stored Applications

#### *12.2.6.1 Removal of Stored Applications on change of MSO*

During boot-up, an OCAP implementation detects and processes a XAIT. The XAIT contains a privileged_certificate_descriptor that can be used to identify the network to which the Host device is attached. If a privileged_certificate_descriptor previously stored by the implementation does not match the privileged_certificate_descriptor signaled in the XAIT, for instance, if the receiver has been detached from one network and attached to another, the OCAP implementation SHALL remove all of the stored applications, except the host manufacturer applications, before launching the Monitor Application. The difference between a stored and a signaled privileged_certificate_descriptor is detected by binary comparison (i.e., two privileged_certificate_descriptors are different if the order of SHA-1 hash value is different). It is not necessary to validate the certificates indicated by the privileged_certificate_descriptor even if the certificates are stored. The OCAP implementation SHALL store the currently signaled privileged_certificate_descriptor in a secure and persistent manner, such that a comparison can be made after a reboot or power-cycle.

#### *12.2.6.2 Removal of Stored Applications on absence of XAIT*

When a XAIT is not accessible via OOB (e.g., no CableCARD is inserted, connected to a basic OpenCable network etc.), applications SHALL NOT be removed for 24 hours after loss of XAIT signaling, after which they MAY be removed.

### 12.2.7 Authentication of Stored Applications

An OCAP implementation SHALL complete a full authentication process before it launches a signed stored application. The authentication process MAY be performed in multiple stages at different time periods as long as the receiver can establish the validity of the total authentication process at the time that the application is launched. OCAP does not require repetition of previously completed aspects of the authentication process that are unaffected by changes in time or file content. The OCAP implementation MAY validate the hash file, signature file and certificates associated with an application at the time that it is stored and, as long as all authenticated files and associated hash and signature files are stored and remain unchanged, not repeat the validation of the hash file and signature file contents at the time of the launch. The implementation SHALL, at a minimum, re-validate the certificate file(s) at the time that an application is launched to ensure that the certificates are valid at that time and match to a self-signed root certificate currently known to the receiver. Note that, in the case that the application is signed more than once, the certificate chain that is used to authenticate the application MAY not be the same on different invocations. In the case that an application would fail to authenticate on launch, for example, because of certificate expiration, the application provider SHALL install a new version of the application which will correctly authenticate.

Applications that have been launched are not re-authenticated during their life cycle irrespective of the duration of that life cycle.

### 12.2.8 The Application description file

#### *12.2.8.1 Description*

The "Application description file" provides the list of files that need to be installed as well as other related necessary information. The notation uses an XML-based syntax.

The "Application description file" is located in the base directory of the OCAP-J application. It specifies locations of all files and directories to be copied into the host. Taken together the directory and file entries in an ADF entry can be converted to a path defined in absolute path format and which is relative to the directory where the ADF is located in the transmission file system the application is delivered in. The absolute path format is the same as the ocap_abs_path term used in locators; see Section 16.2.1.1.2. For example, an implementation can create an absolute path from the XML sample below using "/com/ocap/App.class". A complete path to an ADF entry in a transmission file system can be created by concatenating the absolute path where the ADF is located in the transmission file system in front of an absolute path created from the ADF entry. For example, if the ADF were located at "/apps" a complete path in the transmission file system to the entry in the sample below would be "/apps/com/ocap/App.class". The OCAP implementation shall provide an application with access to the files and directories described in the ADF in the same storage directory structure in storage as the original structure in the file transmission system. It is allowed that the OCAP implementation store files in a different structure internally as long as it provides the original structure to the application.

For example, the OCAP implementation may create a new additional top directory (or directories) over the top of copied directory structure to separate from another directory structure. In such case, the OCAP implementation SHALL convert original class path information to the new directory structure. For example, "/com/ocap/App.class" in an original DSMCC object carousel may be copied to "/copytop/com/ocap/App.class". In this case, a "/com/ocap" class path in XAIT SHALL be converted to "/copytop/com/ocap". ADF is the following:

```
<applicationdescription>
            <dir name="com">
                <dir name="ocap">
                    <file name="App.class" size="10"/>
                </dir>
            </dir>
</applicationdescription>
```

Note that a class path may be described in XAIT in a format of "base_dir + relative_classpath_extension" or "absolute_classpath_extension".

It should be noted that the "Application description file" does not provide all of the information needed to run the application - the OCAP terminal also needs to use the signaling information when loading the application.

The "Application description file" can use an abstract entry, meaning it contains the wild-card character "*" (0x2A) to indicate that all file objects within the specified directory or sub-directories SHALL be stored. The omission of the "Application description file" is equivalent to using this wild-card character in the root directory of the transport file system for file objects, (i.e., to use the following "Application description file" in the base directory):

```
<applicationdescription>
    <dir name="*"/>
    <file name="*" size="0"/>
  </applicationdescription>
```

**Note:** The size attribute is ignored since file name is a wild card.

When a receiver has not correctly stored one or more of the files explicitly listed (i.e., not stored through a wild-card character) in the "Application description file", then the receiver SHALL NOT launch the application. In this case,

the correct storage of a file is measured both by its existence and by matching its file size with that specified in the "Application description file".

Where a file is listed in the "Application description file" of more than one application and is stored, the implementation SHALL ensure that each application sees the correct version of the file for that application. The version of the file visible to one application SHALL NOT be changed by any changes in the version of the file visible to any other stored application which may share that same file.

**Note:** The authors of application description files are responsible for ensuring that files containing data that needs to be dynamic are not listed in the application description file and for ensuring that applications intended to be stored have been written so that relative paths are not used to access files which contain dynamic data.

#### *12.2.8.2 Application description file name and location*

The application description file SHALL be located in the base directory of a OCAP-J application as defined in the application signaling. By convention, the name of this file is:

```
'ocap.storage.oooooooo.aaaa'
```

where:

oooooooo is the organization_id of the application as a 8 character hexadecimal string aaaa is the application_id as a 4 character hexadecimal string

##### *12.2.8.2.1 Syntax*

The syntax of the "Application description file" is defined by the following XML DTD.

The following formal public identifier SHALL be used to identify the Application Description File DTD:

```
"-//OCAP//DTD Application Description File 1.0//EN"
```

and the following URL for the SystemLiteral may be used to reference this file:

```
"http://www.opencable.com/ocap/dtd/applicationdescriptionfile-1-0.dtd"
```

```
The Name used in the document type declaration shall be "applicationdescription".
    <!-- the main element for the application description -->
    <!ENTITY % object "(dir|file)">
    <!ELEMENT applicationdescription %object;+>
    <!ELEMENT dir %object;*>
    <!ATTLIST dir
        name CDATA #REQUIRED
    >
    <!ELEMENT file EMPTY>
    <!ATTLIST file
        name CDATA #REQUIRED
        size NMTOKEN #REQUIRED
    >
```

##### *12.2.8.2.2 Semantics*

**applicationdescription:** This tag models the abstract file and directory locations from the root of the file transmission system. Each file path component is modeled as a dir tag or, in the case of a leaf component, a file tag. This tag shall specify all files and directories to be copied to storage.

**dir:** This tag specifies a directory that models an abstract location of a directory component or, when the name is the wild card character, set of directory hierarchies, to be copied.

**file:** This tag specifies a file that models an abstract location of a file or, when the name is the wild card character, set of files to be copied.

**name:** This attribute provides the name of a file system object (directory or file) that is storable. This is the name of the object within its enclosing directory and hence does not include any directory path information. A name entry MAY contain characters identified in the pchar term defined in [RFC 2396] section 3.3 Path Component, with the following constraints:

- The name attribute value SHALL NOT contain characters outside of the set defined by the pchar term.
- The name attribute value SHALL NOT be the string "." or the string "..".
- Unless the value of the name attribute is "*", then the substring "*" SHALL NOT appear in the value.

Any characters encoded with the percent "%" character, in order to create an escaped sequence, SHALL be un-escaped prior to evaluation of these constraints. If a name attribute value violates any of these constraints the entire XML element containing the name attribute SHALL be invalidated. When this name is the wild card "*", it implies that all objects of this type in this directory SHALL be stored and, in the case of a directory included through a wild card, that all objects in the hierarchy under this directory SHALL be stored.

**NOTE 1:** No elements are provided for naming object types such as Stream or StreamEvent. Therefore there is no mechanism to specify that Stream and StreamEvent objects are required to be stored.

**NOTE 2**: Listing a directory object in the "Application description file" does not imply anything about the contents of the directory unless they are themselves listed in the "Application description file".

**NOTE 3:** If an ADF XML entry does not have an exact match in the transport, the entry SHALL be disregarded successfully.

**size:** This attribute defines the size in bytes of the file. The value of this attribute SHALL be restricted to a sequence of one or more decimal digit characters, and SHALL NOT include leading zeroes. This attribute is ignored when the file name is a wild card.

# 13 EXECUTION ENGINE PLATFORM

This section contains requirements that are extensions to [DVB-GEM 1.1] Section: 11 DVB-J platform.

The Execution Engine is based on Clause 11, "DVB-J Platform". In addition, this section introduces extensions to the DVB-J Platform (e.g., as described in [DVB-GEM 1.1] clause 4.1.4, "Addition of non-GEM interfaces").

## 13.1 DVB-GEM and DVB-MHP Specification Correspondence (informative)

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section. Section 13 (this section) of the OCAP 1.1 Profile corresponds to Section 11 of [DVB-MHP1.1.2] and [DVB-GEM 1.1] as follows:

***Table 13–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP 1.1 Section | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| 13 Execution Engine Platform | 11 DVB-J platform | Extension |
| 13.1 DVB-GEM and DVB-MHP Specification Correspondence (informative) | No Corresponding Section | OCAP-Specific Extension |
| 13.2 Compliance with GEM DVB-J Platform | No Corresponding Section | OCAP-Specific Extension |
| 13.3 OCAP 1.1 Extensions to GEM | No Corresponding Section | OCAP-Specific Extension |
| 13.4 Full Java API List (informative) | No Corresponding Section | OCAP-Specific Extension |
| 13.5 GEM and | No Corresponding Section | OCAP-Specific Extension |

## 13.2 Compliance with GEM DVB-J Platform

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

This section corresponds to [DVB-GEM 1.1] clause 11.

As a consequence of Section 6.1, the Execution Engine is as specified in [DVB-GEM 1.1] Clause 11, "DVB-J Platform".

## 13.3 OCAP 1.1 Extensions to GEM

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

### 13.3.1 General Issues

#### *13.3.1.1 Basic Considerations*

OCAP 1.1 applications SHALL NOT use additional public or protected methods or fields in the `org.ocap.*` namespace that are not listed in this specification or a published extension to this specification. An extension complies with section 13.3.12 of this specification with a system property 'ocap.api.xxx' defined. Applications that

are designed to scale on multiple revisions of this specification SHALL only use `org.ocap.*` methods or fields appropriate to the version of the platform on which they are running.

OCAP 1.1 applications SHALL NOT define classes or interfaces in any package namespace defined in this `specification. OCAP 1.1 terminals SHALL enforce this using the SecurityManager.checkPackageDefinition` mechanism.

**Note:** (informative) This is consistent with the rules for packages defined by MHP, as specified in [DVB-MHP1.1.2] clause 11.2.1 as modified in [DVB-GEM 1.1] clause 11.2.

#### *13.3.1.2 Class Loading Shared Classes*

This section extends and modifies [DVB-GEM 1.1] clause 11.2, specifically [DVB-MHP1.1.2] clause 11.2.3 Class Loading, and provides a mechanism for sharing class files between applications; see Section 21.2.1.20.

As specified by [DVB-MHP1.1.2] clause 11.2.3, when loading a shared class or authenticated file in a signed application, the class or authenticated file shall be signed by at least one of the certificates used to sign the initial xlet class of the application. In the context of loading shared classes, this requirement SHALL NOT apply to the application that references (uses) the shared classes; rather, it SHALL apply to the application that installs (registers) the shared classes. In the case of loading a shared class or authenticated file that is contained in a shared class library, authentication of the shared class or file SHALL be deemed to have occurred when the shared class library was authenticated at time of registration (installation). Therefore, further authentication of the shared class or file is not required at time of use by a referencing application.

**Note:** The effect of the above paragraph is that signing and authentication of shared classes and files contained in a shared class library is independent from the signing and authentication of non-shared classes and files in a signed application that references the shared library. Since an application that wishes to use a shared class library must nevertheless be granted RegisteredApiUserPermission, the monitor application may exercise control over such usage according to operator policy.

**Note:** A (bound or unbound) application that makes use of a shared class library but does not require being granted some MonitorAppPermission() is expected to make use of an application_id in the range 0x4000 to 0x5fff, indicating it is a signed application without monitor permission. If the application or some referenced shared class does require being granted some MonitorAppPermission(), then it is expected to make use of an application_id in the range 0x6000 to 0x7fff indicating it is a dually signed application with monitor permission. See section 11.2.2.3.1 for further information on application_id values.

#### *13.3.1.3 Event Listeners*

In `org.ocap.*` all methods to remove event listeners SHALL have no effect if the listener is not registered.

**Note:** (informative) This is consistent with the rules for event listeners defined by MHP, as specified in [DVB-MHP1.1.2] clause 11.2.5 which is included in [DVB-GEM 1.1] clause 11.2.

#### *13.3.1.4 Event Model in OCAP APIs*

Each class in `org.ocap.*` inheriting from `java.util.EventObject` is just a container for these fields and no validity checks are done for the parameters by this constructor. Instances of these classes are intended to be constructed by the platform implementation and not by applications. The platform implementation will only construct these events with the appropriate information passed in as defined by the constructor.

In `org.ocap.*` all methods to add event listeners SHALL add each listener only once if the add method is called with the same parameters multiple times. This means that the same event is delivered only once to each listener even if it has been added twice.

**Note:** (informative) This is consistent with the event model for DAVIC and DVB APIs, as specified in [DVB-MHP1.1.2] clause 11.2.7 which is included in [DVB-GEM 1.1] clause 11.2.

#### *13.3.1.5 Tuning as a Side-Effect*

No OCAP 1.1 API SHALL cause tuning unless explicitly specified as such.

**Note:** (informative) This is consistent with the rule for MHP APIs, as specified in [DVB-MHP1.1.2] clause 11.2.8 which is included in [DVB-GEM 1.1] clause 11.2.

### 13.3.2 OCAP Platform APIs

OCAP 1.1 requires the following APIs:

#### *13.3.2.1 org.ocap Package*

OCAP 1.1 terminals SHALL support the `org.ocap` package as defined in Annex O.

#### *13.3.2.2 org.ocap.application Package*

OCAP 1.1 terminals SHALL support the `org.ocap.application` package as defined in Annex G.

#### *13.3.2.3 org.ocap.system Package*

OCAP 1.1 terminals SHALL support the `org.ocap.system` package as defined in Annex Q.

#### *13.3.2.4 org.ocap.event*

OCAP 1.1 terminals SHALL support the org.ocap.event package as defined in Annex K.

The `org.ocap.event.EventManager` instance SHALL behave in the same manner as described for `org.dvb.event.EventManager` with regard to listeners.

**Note:** (informative) This is specified in [DVB-MHP1.1.2] clause 11.3.2.2, which is included in [DVB-GEM 1.1] clause 11.3.

The OCAP EventManager enables the use of event filters by an application with appropriate permissions, such as a Monitor Application.

For OCAP 1.1 the `org.ocap.event.EventManager.getInstance()` method SHALL return the object for the `org.ocap.event.EventManager` class.

OCAP 1.1 terminals SHALL report instances of `org.ocap.ui.event.OCRcEvent` through the normal `java.awt` event mechanism. This is facilitated by the inheritance from `java.awt.event.KeyEvent.`

#### *13.3.2.5 org.ocap.net*

OCAP 1.1 terminals SHALL support the `org.ocap.net` package as defined in Annex I.

OCAP 1.1 terminals SHALL return an instance of org.ocap.net.OcapLocator from javax.tv.locator.LocatorFactory, as discussed in [DVB-GEM 1.1] clause 11.3.

OCAP 1.1 terminals SHALL support the requirements of Annex B.

#### *13.3.2.6 org.ocap.hardware*

OCAP 1.1 terminals SHALL support the `org.ocap.hardware` package as defined in Annex F.

#### *13.3.2.7 org.ocap.hardware.pod*

OCAP 1.1 terminals SHALL support the `org.ocap.hardware.pod` package as defined in Annex R.

#### *13.3.2.8 org.ocap.media*

OCAP 1.1 terminals SHALL support the `org.ocap.media` package as defined in Annex S.

#### *13.3.2.9 org.ocap.resource*

OCAP 1.1 terminals SHALL support the `org.ocap.resource` package as defined in Annex L.

#### *13.3.2.10 org.ocap.service*

OCAP 1.1 terminals SHALL support the `org.ocap.service` package as defined in Annex P.

#### *13.3.2.11 org.ocap.system*

OCAP 1.1 terminals SHALL support the `org.ocap.system` package as defined in Annex Q.

#### *13.3.2.12 org.ocap.ui and org.ocap.ui.event*

OCAP 1.1 terminals SHALL support the `org.ocap.ui and org.ocap.ui.event` packages, as defined in Annex E.

#### *13.3.2.13 org.ocap.si*

OCAP 1.1 terminals SHALL support the `org.ocap.si` package as defined in Annex T.

#### *13.3.2.14 org.ocap.environment*

OCAP 1.1 host devices SHALL support the org.ocap.environment package as defined in Annex Y. On OCAP 1.1 host devices that only support a cable environment, the behavior of the org.ocap.environment package SHALL be as follows;

- select SHALL immediately post a EnvironmentStateChangedEvent with both fromstate and tostate being SELECTED if the defined requirements for throwing an Exception are not met.
- getState SHALL always return SELECTED.
- deselect SHALL always immediately post a EnvironmentStateChangedEvent with both fromstate and tostate being SELECTED if the defined requirements for throwing an Exception are not met.
- applications MAY register and unregister EnvironmentListeners.

### 13.3.3 OCAP Specific Network APIs

OCAP 1.1 terminals SHALL support the requirements of Annex B.

### 13.3.4 Monitor Application Support

Some of the classes and methods in the OCAP Packages above are provided only for Monitor Application. Access to these APIs requires the Monitor Application permission. OCAP 1.1 terminals SHALL support the requirements of Section 21.

### 13.3.5 Presentation APIs

The Presentation APIs are those interfaces which support the presentation of mixed media and graphical user interfaces in an OCAP 1.1 application.

#### *13.3.5.1 Graphical User Interface API (informative)*

The graphical user interface API is comprised of a combination of Java packages and extensions.

The following packages are referenced:

- `java.awt`, as specified by [DVB-MHP1.1.2] clause 11.4.1, which is included in [DVB-GEM 1.1] clause 11.4.
- `org.havi.ui` and `org.havi.ui.event`, as specified by [DVB-MHP1.1.2] clause 11.4.1.2, which is included in [DVB-GEM 1.1] clause 11.4.
- Parts of various sub-packages of `javax.tv`, as specified by [DVB-MHP1.1.2] clause 11.4.1.2, which is included in [DVB-GEM 1.1] clause 11.4.
- `org.dvb.ui`, as specified by [DVB-MHP1.1.2] clause 11.4.1.3, which is included in [DVB-GEM 1.1] clause 11.4.
- org.ocap.ui, as specified in this specification; see Annex D.

#### *13.3.5.2 Handling of Input Events*

The OCAP 1.1 Profile extends Section 11.4.1.4 in the [DVB-GEM 1.1].

**Note:** (informative) The OCAP 1.1 user input event package, `org.ocap.event`, extends the `org.dvb.event` package. This is specified in Section 13.3.2.4.

DVB-MHP 1.0.3 [DVB-MHP1.1.2] defines the concept of a "resident navigator", which is incorporated into [DVB-GEM 1.1]. OCAP 1.1 does not directly support this concept; instead, an OCAP terminal supports baseline "Watch TV" functionality (see Section 20) and supports OCAP 1.1 applications that embody the functionality of the "resident navigator". OCAP 1.1 applications MAY consume input events by virtue of having AWT focus or by requesting them. The Monitor Application MAY register itself to receive input events prior to reception by other OCAP 1.1 applications and MAY consume input events without passing them on to other OCAP 1.1 applications (see Section 21.2.1.13 and Annex K.2.1).

To ensure a consistent user experience, the following rules are defined:

- An application creating an `HScene` and placing components into it SHALL not, by default, get the input focus for these components.
- The application MAY request to get the input focus by calling `Component.requestFocus()`. If this is granted and the focus moved to the requested component, this component SHALL receive input events as defined in Annex K.
- The application MAY request to receive a subset of input events via the `org.ocap.event` API even when not having the AWT focus.

These rules are consistent with [DVB-GEM 1.1].

#### *13.3.5.2.1 Application recommendations for VK_TELETEXT (informative)*

OCAP 1.1 does not require support for the VK_TELETEXT input event. Thus, some of the application recommendations in [DVB-MHP1.1.2] clause 11.4.1.4 may not be applicable. OCAP 1.1 applications should not rely on the VK_TELETEXT key.

### 13.3.6 Streamed Media API

#### *13.3.6.1 Overview (informative)*

The streamed media APIs provide a consistent interface for streaming digital content such as audio, video, and ancillary data. The API is derived from [JMF] and [Java TV]. The following extended APIs are also supported:

- `org.davic.media`, as specified by [DVB-MHP1.1.2] clause 11.4.2.5.2 as included in and modified by [DVB-GEM 1.1] clause 11.4.
- `org.dvb.media`, as specified by [DVB-MHP1.1.2] clause 11.4.2.5.1 as included in and modified by [DVB-GEM 1.1] clause 11.4.

The language pertaining to `DVBLocator` in [DVB-MHP1.1.2] clause 11.4.2.2 is modified by [DVB-GEM 1.1] clause 11.4; hence, these modifications apply to OCAP 1.1. Specifically, Section 13.3.2.5 introduces `OCAPLocator,` which takes the place of `DVBLocator.`

Classes related to subtitling required in [DVB-MHP1.1.2] clause 11.4.2.5.1 are made optional by [DVB-GEM 1.1] clause 11.4; hence, they are optional in OCAP 1.1.

#### *13.3.6.2 OCAP Extensions to the Framework*

##### *13.3.6.2.1 Additional constraints on Streamed Media API Extensions*

In addition to the requirements of [DVB-GEM 1.1], OCAP 1.1 terminals SHALL obey the extended semantic requirements specified by Annex M of this specification.

**Note:** (informative) This is in addition to the extensions to the framework required by [DVB-MHP1.1.2] clause 11.4.2.5, as included in and modified by [DVB-GEM 1.1], clause 11.4. These modifications include removing the requirement for classes related to subtitling and the DVB CA system.

### 13.3.7 Data Access APIs

**Note:** (informative) Data Access APIs are specified in [DVB-MHP1.1.2], clause 11.5, as included in and modified by [DVB-GEM 1.1], clause 11.5.

OCAP 1.0 extends DVB-GEM 1.0.2 [8] and DVB-MHP 1.0.3 [9] clause 11.5.6 as follows:

- When a running application needs additional room in persistent storage the implementation SHALL NOT remove files owned by applications with a higher application priority than the running application requesting the storage. This requirement alters the file priority hierarchy defined by MHP so that owning application priority is considered before file priority for purposes of unsolicited implementation deletion of files.
- The implementation SHALL NOT expose files owned by unbound network applications in a user interface that allows the files to be selected for removal.
- The implementation SHALL NOT decrease the amount of total storage available in the persistent storage area indicated by the dvb.persistent.root property. An application can discover this value using the org.ocap.storage.StorageManager.getTotalPersistentStorage method.

- For memory management purposes the level of storage that triggers possible removal of files is 90% of the total available persistent storage. In the absence of application request of storage, the implementation SHALL NOT remove files from persistent storage until this level is reached. Files owned by unbound network applications SHALL NOT be removed due to memory management in the absence of application storage requests. OCAP 1.1 terminals SHALL implement the requirements of [DVB-MHP1.1.2], Annex P. These fulfill the abstract requirements described in [DVB-GEM 1.1], Annex P.

#### *13.3.7.1 Overview (informative)*

The Data Access APIs provide an interface for accessing files which are encapsulated in an object carousel or are accessible through a DSM-CC interactive network. The Data Access APIs also provide a consistent interface for persistent storage which may be in the form of a local hard drive, floppy disk, or non-volatile RAM (NVRAM).

The Java packages include:

- `java.net`, as specified by [DVB-MHP1.1.2] Section 11.3.1.6 "java.net" which is included in [DVB-GEM 1.1] clause 11.3.
- `javax.tv.net`, as specified by [DVB-MHP1.1.2] clause 11.5.2 which is included in [DVB-GEM 1.1] clause 11.5.
- `org.dvb.dsmcc`, as specified by [DVB-MHP1.1.2] clause 11.5.1 as included in and modified by [DVB-GEM 1.1] clause 11.5, with the further semantics required by [DVB-MHP1.1.2] annex P.
- `org.dvb.io.persistent`, as specified by [DVB-MHP1.1.2] as included in and modified by [DVB-GEM 1.1].

#### *13.3.7.2 Broadcast Transport Protocol Access API*

The OCAP 1.1 implementation SHALL differentiate the base directory for each OCAP 1.1 application. When an application is signaled via the XAIT or AIT, the OCAP 1.1 implementation SHALL place the base directory in the Java property `ocap.j.location`. The base directory can then be discovered by the application using the `AppAttributes.getProperty()` method.

**Note:** (informative) For applications signaled via the AIT, applications may also discover the base directory using the property `ocap.j.location` and the `AppAttributes.getProperty()` method. This behavior is required by [DVB-GEM 1.1] clause 11.7.2.

When an application that is run from storage uses an instance of org.dvb.dsmcc.DSMCCObject to reference stored files, the following semantics SHALL apply to the methods on that class:

- abort() - SHALL always throw a NothingToAbortException
- asynchronousLoad(AsynchronousLoadingEventListener) - if the file exists, SHALL succeed immediately (with SuccessEvent being generated) otherwise SHALL fail with an InvalidPathNameException
- isObjectKindKnown() - SHALL always returns true
- isStream() - always returns false as a consequence of clause 12.2.8.2.2 Semantics not defining mechanisms to name Stream and StreamEvent objects.
- isStreamEvent() - always returns false as a consequence of clause 12.2.8.2.2 Semantics not defining mechanisms to name Stream and StreamEvent objects.
- loadDirectoryEntry(AsynchronousLoadingEventListener) - SHALL always succeed immediately with SuccessEvent being generated
- prefetch(DSMCCObject,String,byte) and prefetch(String,byte) - SHALL always returns false
- setRetrievalMode(int) - SHALL be silently ignored

- synchronousLoad() - if the file exists, SHALL succeed immediately otherwise SHALL fail with InvalidPathnameException
- getURL() - returns a java.net.URL identifying the stored file in the file-system of the OCAP host device

Additionally the following SHALL apply:

- the state machine of a DSMCCObject is unchanged however it SHALL refer to loading purely from storage and not from the mechanism by which the file was originally distributed
- The unload() method SHALL not be considered as removing stored files from where they are stored
- ObjectChangeEvents SHALL never be generated.
- such DSMCCObjects SHALL always be considered to form part of an attached service domain. Hence methods on DSMCCObject that are defined to fail if the service domain isn't in an attached state SHALL not fail for this reason.

#### *13.3.7.3 Support for IP over the Return Channel (informative)*

Support for IP over the return channel is as specified in [DVB-MHP1.1.2] clause 11.5.3 as included in [DVB-GEM 1.1], clause 11.5. See also the notes in Section 13.5.4 of this profile (OCAP 1.1).

OCAP 1.1 supports `org.ocap.net` package, as specified in Section 13.3.2.5. See also Section 13.3.13.1.

#### *13.3.7.4 MPEG-2 Section Filter API*

**Note:** (informative) Portions of this API are specified by [DVB-MHP1.1.2] clause 11.5.4, as included in [DVB-GEM 1.1] clause 11.5. As required by MHP, the errata in [DVB-MHP1.1.2] Annex A SHALL be applied.

OCAP introduces the following additional APIs and extensions:

In OCAP, the DAVIC MPEG-2 Section Filter API defined in Annex E of [DAVIC] supports section filtering for both an inband transport stream and the Extended Channel. All occurrences of "transport stream" in Annex E of [DAVIC] shall be replaced with "transport stream or the Extended Channel defined in [CCIF 2.0]". The DAVIC section filter is a logical entity and the OCAP implementation associates it with an actual hardware/software section filter in an implementation dependent manner. For example, one actual section filter can be associated with several logical section filters if they filter sections on the same PID. This optimization does not modify the requirements of SectionFilterGroup. Once a group has been successfully reserved, the implementation SHALL be able to find the actual filters necessary if the logical section filters are later changed to be on different PIDs. Note that the OCAP Resource Management system described Section 19 manages the logical section filters associated by the implementation with an org.davic.mpeg.sections.SectionFilterGroup.

If an OCAP-J application wants to filter sections coming from the inband transport stream, it specifies an instance of org.davic.mpeg.TransportStream to the SectionFilterGroup.attach() method. The section filters are connected to the inband transport stream. The behavior is as described in the Annex E of [DAVIC] with the following extension:

- The OCAP implementation shall descramble CA automatically according to Section 16.2.1.6. If the OCAP implementation terminates CA descrambling of the filtering section stream, the current section filter stops with notification of the EndOfFilteringEvent. Note that the CA resource is not managed by the OCAP Resource Management system described in Section 19.

If an OCAP-J application wants to filter sections coming from the Extended Channel defined in [CCIF 2.0], it specifies an instance of org.ocap.mpeg.PODExtendedChannel to the org.davic.mpeg.sections.SectionFilterGroup.attach() method. The org.ocap.mpeg.PODExtendedChannel is a sub class of the org.davic.mpeg.TransportStream. The behavior is completely the same as described in Annex E of [DAVIC], except for the following:

- The section filters are connected to the Extended Channel so they can retrieve specific section(s) coming from the MPEG section flow of the Extended Channel.
- It depends on the CableCARD implementation, which mode is used from three modes (the OOB mode, the DSG mode, or the DSG one-way mode). The CableCARD selects the mode via a set_dsg_mode() APDU. (The DSG or DSG one-way mode can be selected only if the Host supports it.) In either mode, if necessary, the Host sends a new_flow_req() APDU with a MPEG_section service type to open a MPEG section flow, to retrieve sections from the Extended Channel. The CableCARD sends a new_flow_req() to open a DSG flow only if in the DSG or DSG one-way mode. See also [CCIF 2.0] for details of flow management.
- Note that it is implementation specific whether several logical section filters can share one MPEG section flow if they filter sections of the same PID. The MPEG section flow is not managed by the OCAP Resource Management system in Section 19. When filtering is finished, the OCAP implementation can keep the MPEG section flow open for the future section filtering if it doesn't prevent other section filtering.
- The OCAP implementation SHALL notify applications of the termination of filtering using the section filter events. If the startFiltering() method is called, the OCAP implementation SHALL open a new MPEG section flow if the previous MPEG section flow has already closed.
- The org.davic.mpeg.sections.SectionFilterGroup.attach() method throws the FilterResourceException, if reserving a necessary flow on the Extended Channel fails.
- A series of calls to org.davic.mpeg.sections.SectionFilter.startFiltering() throw the FilterResourceException, if they can't open the flow for the specified PID.

For both inband and OOB section filtering, including every mode (the OOB mode, the DSG mode and the DSG one way mode), the section filter events SHALL be delivered as specified in the Annex E of [DAVIC].

#### *13.3.7.5 Persistent Storage API*

This section extends [DVB-MHP1.1.2] clause 11, which is included by [DVB-GEM 1.1] clause 11. This section adds a storage management API that enables multiple storage devices of various types to be supported by the implementation; see Annex V. The ExtendedFileAccessPermissions extends org.dvb.io.persistent.FileAccessPermissions and everywhere that FileAccessPermissions is referenced ExtendedFileAccessPermissions SHALL be used instead. An org.ocap.storage.StorageProxy corresponds to a single storage device. Each persistent storage device contained within or attached to a Host device SHALL be represented by a StorageProxy, unless the device is reserved 100% by the implementation for storage of implementation software and supporting data files. In addition, the implementation SHOULD NOT create a StorageProxy for the persistent storage device containing directory indicated by the dvb.persistent.root property. Implementations SHALL support discovery of all StorageProxy instances through the org.ocap.storage.StorageManager singleton. The implementation SHALL maintain the integrity of stored content and, if necessary, restore that integrity at boot or attachment time. If a device was removed or turned off during operation, only content being written at that time SHOULD be lost or damaged. The implementation SHALL delete any content whose name or directory information has been lost.

Each StorageProxy contains one or more logical volumes, each represented by an org.ocap.storage.LogicalStorageVolume. A LogicalStorageVolume is not equivalent to a partition. In its basic form, a LogicalStorageVolume is similar to a directory and may be used by an application to organize content. A LogicalStorageVolume represents a directory structure that an application is allowed to access. The application can only access the directory returned from the LogicalStorageVolume.getPath() method and its sub-directories. The interface is also intended as a basis for extensions that may expose specialized capabilities of some storage architectures, such as storage pre allocation and exclusive use for video storage. OCAP 1.1 supports a general purpose storage volume to be used for data file formats supported by OCAP. A general purpose storage volume SHALL meet persistent storage requirements for file type support. Other volume types are out-of-scope in the OCAP 1.1 profile and may be defined by other specifications.

Implementations SHALL support the allocation of multiple LogicalStorageVolume instances within a StorageProxy by applications granted persistent storage permission via the "file" element in their permission request files. Each LogicalStorageVolume also has an org.ocap.storage.ExtendedFileAccessPermissions assigned to it. The ExtendedFileAccessPermissions extends org.dvb.io.persistent.FileAccessPermissions to allow applications with different organization identifiers to have read and write permission for the LogicalStorageVolume. In addition, any file or directory in persistent storage MAY be given ExtendedFileAccessPermissions via the org.dvb.io.persistent.FileAttributes class. This means that in OCAP 1.1 the org.dvb.io.persistent.FileAttributes class SHALL support instances of the org.ocap.storage.ExtendedFileAccessPermissions class when passed in place of an org.dvb.io.persistent.FileAccessPermissions, assign the permissions per class definition of the parameter instance, and return instances of the parameter type from the getFileAccessPermissions() method. Note that an application can access files and directories via java.io package without obtaining a LogicalStorageVolume instance. Other than the extended permissions, all of the requirements specified for java.io and org.dvb.io.persistent packages SHALL apply.

As an implementation created and owned directory, the directory indicated by the dvb.persistent.root property SHALL NOT be exposed by a LogicalStorageVolume.

The implementation SHALL NOT allow applications to access files or directories in persistent storage unless the application has persistent storage access granted by the permission request file "file" element, and access rights to the file or directory. When an application is launched and has persistent storage access granted, the implementation SHALL create necessary permissions for the application for any potential LogicalStorageVolume allocation the application might make. Potential allocations include those for devices that might be attached after an application with access privileges is launched. Whenever a LogicalStorageVolume is allocated by an application the LogicalStorageVolume.getPath() method SHALL reflect the implementation created directory. This directory SHALL have the form <StorageProxyDesignator>":"/OCAP_LSV/"<orgID>"/"<appId>"/"<LSV_Name>"" where StorageProxyDesignator is the designator for a storage device, LSV_name is the last directory in the path as returned by the getPath method, orgId is the organization Id and appId is the application Id of the allocating application. See [DVB-MHP1.1.2], Section 10.5.1 for organization and application Id formats. If the StorageProxyDesignator is not specified in the path the implementation will provide a StorageProxyDesignator corresponding to a default storage device. If LSV_Name is not specified the implementation will use LSV_appID. The "/OCAP_LSV/" directory SHALL be reserved for LogicalStorageVolume allocation. Applications will be granted read only access to this base directory. The implementation SHALL NOT allow an application to create a LogicalStorageVolume with a path that is identical to an existing LogicalStorageVolume instance or the value in dvb.persistent.root property. When allocating, an application will set the access permissions of the LogicalVolumeStorage for access by the application, organization, organizations different from the application setting the access, or world.

If the implementation is able to determine that the storage device is initialized and empty, it MAY initialize it. If the implementation determines that the device is not currently initialized for use, and the implementation is unable to determine that the device is empty, it SHOULD indicate through the StorageProxy APIs that the device is present but contains an unsupported format via the UNSUPPORTED_FORMAT state and therefore needs to be initialized before use. In this case, it would be the responsibility of an OCAP application to notify the user of the potential loss of data and request that implementation initialize the device.

#### *13.3.7.5.1 Re-creation of LogicalStorageVolume Instances*

When a Host device is power-cycled or rebooted the implementation SHALL re-create LogicalStorageVolume instances to a state identical to that which existed before the power-cycle or reboot. The path and file attributes including priority, expiration date, and extended file access permissions SHALL be identical in this case.

When a storage device corresponding to a StorageProxy instance containing LogicalStorageVolume instances is moved to a different Host device, the implementation in the new Host device SHALL re-create LogicalStorageVolume instances such that the paths are identical. If the new Host device cannot re-create file attributes including extended file access permissions, priority, and expiration date, it SHALL set the file attributes to default values. In this case a LogicalStorageVolume SHALL be re-created for each path discovered in the storage

device with a name matching the format <StorageProxyDesignator>":"/OCAP_LSV/"<orgID>"/"<appId>"/"<LSV_Name>"", where StorageProxyDesignator is the designator for a storage device, LSV_name is the last directory in the path as returned by the getPath method, orgId is an organization identifier and appId is an application Id as defined in [DVB-MHP1.1.2], Section 10.5.1. LogicalStorageVolume instances created in this fashion SHALL be setup with the default extended file access permissions readApplicationRight, writeApplicationRight, readOrganizationRight. When using default values other file attributes are set to default values as defined by the FileAttributes class description; see [DVB-MHP1.1.2], Annex K. Applications with organization and application Id's matching a LogicalStorageVolume instance's path orgId and appId SHALL be given applicable application rights and applications with matching organization Id's SHALL be given applicable organization rights.

#### *13.3.7.5.2 Detachable Devices*

This specification allows an implementation to support detachable (e.g., external or hot-pluggable) devices and requires certain behavior of implementations supporting detachable devices. When a storage device is attached, an implementation supporting detachable devices SHALL add a corresponding StorageProxy to the set maintained by the StorageManager. This StorageProxy SHALL provide a DetachableStorageOption. Implementations SHALL determine whether a newly discovered device is supported. If the device is unsupported, the corresponding StorageProxy SHALL place the StorageProxy in the UNSUPPORTED_DEVICE state. If the device is supported, the implementation SHALL determine whether it is already initialized for use with the implementation. If the device has already been initialized for use with the implementation, the implementation SHALL make the volumes for the StorageProxy available for use as soon as possible without any application involvement and place the StorageProxy in the READY state.

An application can invoke the makeDetachable() method on the DetachableStorageContext to request that the implementation make a device safe to detach. The effects are implementation- and device-dependent, but may include the closing of files, flushing of buffers, and the removal of power from the device. If the implementation succeeds in making the device safe for removal, the implementation MAY remove the corresponding StorageProxy from the registry or it MAY place the StorageProxy in the OFFLINE state. When the makeReady() method is invoked on a StorageProxy in the OFFLINE state, an implementation SHOULD attempt to reactivate the device and place it in the appropriate state as if it were newly attached.

#### *13.3.7.5.3 Removable Devices*

Certain storage devices have a permanently installed bay that the storage medium can be removed from while power is asserted, e.g., memory stick. These differ from detachable devices which have a connector but no permanent bay. Some devices MAY be detachable and removable, e.g., IEEE-1394 connected memory stick bay. A RemovableStorageOption is included as a basic interface for removable devices and SHALL be contained by any StorageProxy that represents a removable storage device.

#### *13.3.7.5.4 Implementation Provided LogicalStorageVolume*

Whenever a detachable or removable storage device is connected to a Host device and the storage device is pre-formatted and contains data files that are not under the "/OCAP_1-0" directory or any of its sub-directories, the implementation SHALL create a LogicalStorageVolume that provides access to all of those files. Such LogicalStorageVolume instances SHALL have the following attributes:

- "The absolute path of the volume is the top level directory on the device.
- "The file access permissions have readWorldAccessRight and readApplicationAccessRight true, otherOrganizationsWriteAccessRights and otherOrganizationsReadAccessRights of zero length and all other permissions false.
- "The owner is the implementation and hence the LogicalStorageVolume.setFileAccessPermissions method SHALL throw SecurityException. Calling the static org.dvb.io.persistent.FileAttributes.setFileAttributes

method for any file in such a LogicalStorageVolume SHALL throw SecurityException unless the storage device is read-only due to hardware constraints, in which case this method SHALL throw IOException.

INFORMATIVE NOTE: As a consequence, all files in an implementation provided LogicalStorageVolume are readable by all OCAP-J applications.

#### *13.3.7.5.5 Extended File Access Permissions Support*

Detachable or removable storage devices may have been initialized by a non-OCAP device in a format which is supported by the implementation but which does not support the full semantics of either org.ocap.storage.ExtendedFileAccessPermissions or org.dvb.io.persistent.FileAccessPermissions. For example, a memory stick initialized externally might not support organization or application write access. In this case the StorageProxy.allocateGeneralPurposeVolume MAY fail when passed an ExtendedFileAccessPermission parameter the storage device does not support. The StorageProxy.getSupportedAccessRights method indicates which permissions are supported by the storage device.

### 13.3.8 Service Information and Selection APIs (informative)

These APIs are specified in [DVB-GEM 1.1] clause 11.6.

Service Information APIs provide an application with a mechanism for finding out information about available services in an interactive broadcast environment. These services may include Electronic Programming Guides (EPGs), Video-On-Demand (VOD), and Enhanced Broadcasting.

The Service Selection APIs provide an interface for selecting these services for presentation.

### 13.3.9 Common Infrastructure APIs (informative)

In addition to the requirements of [DVB-GEM 1.1] clause 11.7, Section 13.3.2.11, org.ocap.system requires support for the package `org.ocap.system`, as specified in Annex Q.

**Note:** (informative) [DVB-GEM 1.1] clause 11.7 specifies APIs in the packages javax.tv.xlet, java.rmi, and org.dvb.application.

### 13.3.10 Termination of OCAP-J Applications by the Platform

The platform SHALL cease execution of an application after it has been destroyed. An application is considered destroyed when its `destroyXlet()` method has been invoked with the unconditional flag set to `true`, the `destroyXlet()` method returns successfully, or the application manager has otherwise transitioned the application to the `DESTROYED` state.

When an application is destroyed, the implementation SHALL ensure that:

- Network Interface resources are handled as defined in section 19.2.1.1.5.2.
- All other resources held by a destroyed application are recovered when the application is terminated.
- All execution stacks are safely unwound.
- The application ceases to execute.
- Applications running at the same time are not interrupted or restarted.

**Note:** (Informative): There may be circumstances in which an application is in a non-responsive state and a call to `Xlet.destroyXlet()` does not execute, or the `XletContext.notifyDestroyed()` method is not invoked. A non-exclusive list of such circumstances includes the following:

- An application being in an infinite loop

- An application catching java.lang.ThreadDeath
- Applications using finalizers to create new instance and starting a new Thread running the `run()` method of the new instance.
- Applications waiting in blocking I/O calls (e.g., setup a `ServerSocket` with no timeout, and setting it to await a connection, which never happens.)

These requirements depend in part upon JVM technology not currently widely available. For background, see [JVM Tech] and [JSR].

**Note:** (informative) The platform requirements of this section are related to the requirements on applications in [DVB-MHP1.1.2] clause 11.7.1.2, as included in [DVB-GEM 1.1] clause 11.7. Essentially, the above ensures platform stability when applications fail to follow the rules.

### 13.3.11 Application Discovery and Launching APIs (informative)

In addition to the APIs as specified by [DVB-MHP1.1.2] clause 11.7.2, as included in [DVB-GEM 1.1] clause 11.7.2, the `org.dvb.application.AppAttributes` interface is extended with the `org.ocap.application.OcapAppAttributes` interface, as required by Section 13.3.2.2.

### 13.3.12 Profile and Version Properties

In addition to the profile and version properties specified by [DVB-GEM 1.1] clause 11.9.3, OCAP reserves property names beginning the with string 'ocap'. OCAP terminals SHALL support the Profile and Version properties as per Table 13–2:

***Table 13–2 - Profile and version properties***

| Property | Semantics | Possible Values |
|---|---|---|
| ocap.profile | Name of the profile of the Host device. | "OCAP 1.1" |
| ocap.version | Latest version of the OCAP profile supported by the Host device. | "1.0" |

#### *13.3.12.1 Information on Options (informative)*

DVB-MHP 1.0.3 [DVB-MHP1.1.2] clause 11.9.3.1, as included in [DVB-GEM 1.1] clause 11.9.3, defines properties for optional features. As with MHP and GEM, OCAP 1.1 does not require any of the optional features.

#### *13.3.12.2 Optional API Support*

In addition to optional properties specified by [DVB-MHP1.1.2], Section 11.9.3, OCAP adds the ability for an implementation to indicate support for APIs that extend OCAP. API specifications that extend OCAP SHALL specify a unique name to be appended to the reserved string name, 'ocap.api.option' (e.g., "ocap.api.option.dvr"). The property SHALL contain the version of the API, using a string specified by releases of the corresponding API specification.

#### *13.3.12.3 System Properties*

This section extends [DVB-GEM 1.1] Section 11.9.3.

OCAP 1.1 defines the following system properties as per Table 13–3:

***Table 13–3 - System Properties***

| Property | Description | Value | Application Access |
|---|---|---|---|
| ocap.hardware.vendor_id | vendor id- the Organizationally Unique Identifier (OUI) for the manufacturer. (See Table 9.19-5, [CCIF 2.0]) | Int<br>Range: 0<value<224 | MonitorAppPermission("properties") |
| ocap.hardware.version_id | Unique Hardware identifier assigned to each version of hardware from a particular vendor. (See Table 9.19-5 of [CCIF 2.0] | Long<br>Range: 0<value<232 | MonitorAppPermission("properties") |
| ocap.hardware.createdate | manufacture date, formatted as "MM-DD-YYYY | string | MonitorAppPermission("properties") |
| ocap.hardware.serialnum | serial number | string | MonitorAppPermission("properties") |
| ocap.memory.video | separate video memory in KB | Long Range: 0<value | MonitorAppPermission("properties") |
| ocap.memory.total | total DRAM in KB - represents footprint of device and not memory available to applications | Long Range: 0<value | MonitorAppPermission("properties") |
| ocap.system.highdef | support for high definition | boolean | unsigned and signed |

### 13.3.13 Java Permissions

#### *13.3.13.1 org.dvb.net.rc.RCPermission (informative)*

[DVB-MHP1.1.2] clause 11.10.2.5, as included in [DVB-GEM 1.1] clause 11.10 defines the class `org.dvb.net.rc.RCPermission`. OCAP terminals are required to support the class as specified in MHP, however, because OCAP terminals will not have dial-up modems, the string "target:" + the phone number will be of little use. Applications that attempt to dial the modem using the `org.dvb.net.rc.RCInterface` class will fail as specified for this API.

#### *13.3.13.2 java.io.FilePermission*

When an unsigned/signed OCAP application runs from storage, a java.io.FilePermission with action "read" shall be granted for the root directory of the original file transmission system, e.g., DSMCC object carousel, and the root directory for stored files according to the ADF and referenced SCDF and (recursively) all files and subdirectories contained in those directories. Accessing any files or directories above any of these root directories shall throw a SecurityException as defined in the specification for the java.io package. For example, an additional top directory above the copied directory structure according to ADF or SCDF is not accessible. (See also Section 12.2.8.1 and Section 21.2.1.21.1.)

#### *13.3.13.3 java.util.PropertyPermission*

This section extends [DVB-GEM 1.1] Section 11.10, which refers to [DVB-MHP1.1.2] Section 11.10.2.1.

For all applications, a read permission shall be granted for all properties defined in Section 13.3.12.3 identified as available for unsigned applications. The permission shall be denied for the action string "write".

See also Section 21.2.1.20 for details of properties only available to the Monitor Application.

#### *13.3.13.4 java.net.SocketPermission*

OCAP extends [DVB-MHP1.1.2], Section 11.10.2.9 as included by [DVB-GEM 1.1], Section 11.10 and requires granting of SocketPermission for local host access by default. When a signed or unsigned application is being launched java.net.SocketPermission("localhost", "accept, connect, listen") SHALL be granted to the application.

## 13.4 Full Java API List (informative)

This subsection does not correspond to any [DVB-GEM 1.1] Section.

Section 11 in [DVB-GEM 1.1] does require, with some modification, the Java packages that are detailed in [DVB-MHP1.1.2]. Therefore, these packages are included in OCAP by default. This subsection summarizes those packages that are included in an implementation of the OpenCable Application Platform.

**Note:** See [DVB-MHP1.1.2] as included in and modified by [DVB-GEM 1.1] for the definitive Core Java packages, including JavaTV and JMF. For all packages listed below, please refer to the normative specification(s) of each.

### 13.4.1 Java Platform Packages

```
java.awt
java.awt.color
java.awt.event
java.awt.font
java.awt.im
java.awt.image
java.beans
java.io
java.lang
java.lang.ref
java.lang.reflect
java.math
javax.microedition.xlet
javax.microedition.xlet.ixc
java.net
java.rmi.registry
java.rmi
java.security
java.security.acl
java.security.cert
java.security.interfaces
java.security.spec
java.text
java.util
java.util.jar
java.util.zip
javax.microedition.io
javax.microedition.pki
javax.security.auth.x500
```

### 13.4.2 Java TV 1.0 Packages

```
javax.tv.graphics
javax.tv.locator
javax.tv.media
javax.tv.net
javax.tv.service
javax.tv.service.guide
javax.tv.service.navigation
javax.tv.service.selection
javax.tv.service.transport
javax.tv.util
javax.tv.xlet
```

**Note:** `javax.tv.carousel` and `javax.tv.media.protocol` have been excluded, as defined by [DVB-MHP1.1.2].

### 13.4.3 Java Media Framework 1.0 Packages

```
javax.media
javax.media.protocol
```

### 13.4.4 Java Secure Socket Extension 1.02 Packages

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

### 13.4.5 HAVi Level 2 User Interface Packages

```
org.havi.ui
org.havi.ui.event
```

### 13.4.6 DVB-MHP 1.1.2 Packages

The following list of packages required by OCAP excludes `org.dvb.si` from the [DVB-MHP1.1.2] list of packages:

```
org.dvb.application
org.dvb.dsmcc
org.dvb.event
org.dvb.io.ixc
org.dvb.io.persistent
org.dvb.lang
org.dvb.media
org.dvb.net
org.dvb.net.rc
org.dvb.net.tuning (except for DvbNetworkInterfaceSIUtil class)
org.dvb.test
org.dvb.ui
org.dvb.user
```

### 13.4.7 DAVIC 1.4.1, Part 9 Packages

```
org.davic.media
org.davic.mpeg
org.davic.mpeg.sections
org.davic.net
org.davic.net.tuning
org.davic.resources
```

### 13.4.8 OCAP Packages

```
org.ocap (see Annex O)
org.ocap.application (see Annex G)
org.ocap.event (see Annex K)
org.ocap.hardware (see Annex F)
org.ocap.hardware.pod (see Annex R)
org.ocap.media (see Annex S)
org.ocap.net (see Annex I)
org.ocap.resource (see Annex L)
org.ocap.service (see Annex P)
org.ocap.system (see Annex Q)
org.ocap.ui (see Annex E)
org.ocap.ui.event (see Annex E)
org.ocap.si (see Annex T)
org.ocap.mpeg (see Annex H)
org.ocap.storage (see Annex V)
org.ocap.system.event (see Annex U)
org.ocap.test (see Annex W)
org.ocap.dpi (see Annex X)
org.ocap.environment (see Annex Y)
org.ocap.diagnostics (see Annex Z)
```

## 13.5 GEM and OCAP 1.1

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

### 13.5.1 VK_TELETEXT Key

OCAP 1.1 terminals MAY support the VK_TELETEXT input event; support for this input event is not required.

**Note:** (informative) This means that the VK_TELETEXT entry in [DVB-MHP1.1.2] clause G.5 does not apply to OCAP 1.1. See also [DVB-GEM 1.1] clause G as modified by Section 6.1.1.1, and [DVB-MHP1.1.2] clause 11.4.1.4, which is included in [DVB-GEM 1.1] clause 11.4.

### 13.5.2 Factory Methods

#### *13.5.2.1 org.dvb.net.rc.RCInterfaceManager.getInterface*

The `org.dvb.net.rc.RCInterfaceManager.getInterface()` method SHALL return an instance of `org.ocap.net.OCRCInterface`. This is consistent with the API as specified by [DVB-GEM 1.1]

### 13.5.3 Locale Support (informative)

The OCAP 1.1 Profile requires support for locales in addition to those required by [DVB-GEM 1.1]. Locales supported in OCAP 1.1 (this profile) are specified in Section 17.2.1.2.

**Note:** [DVB-MHP1.1.2] clause 11.3.1.3, which is included in [DVB-GEM 1.1] clause 11.3, states that the constants in the class `java.util.Locale` do not imply support (or otherwise) for all locales for which constants are defined. Even with the additional locales required by OCAP 1.1, this informative statement continues to be true.

### 13.5.4 org.dvb.event Package

The org.dvb.event.EventManager.getInstance() method shall return an instance of the org.ocap.event.EventManager that is a subclass of the org.dvb.event.EventManager.

Informative: OCAP 1.1 uses `org.ocap.event.EventManager` which extends `org.dvb.event.EventManager`.

The OCAP 1.1 user input event API as specified in Annex K works through the org.dvb.event package. The OCAP 1.1 package is `org.ocap.event`.

**Note:** `org.dvb.event` is as specified in [DVB-MHP1.1.2] clause 11.3.2.2, which is included in [DVB-GEM 1.1] clause 11.3.

# 14 SECURITY

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 12 Security and are extensions of [DVB-MHP1.1.2] Section: 12 Security.

This OCAP 1.1 chapter indicates in which ways security on an OCAP 1.1 platform conforms to security on a [DVB-MHP1.1.2] platform. It also highlights differences in security mechanism and policy between the two platforms.

The OCAP security structure defined in the remainder of this section is considered an "Extension to MHP Signing Framework" as allowed by Section 12.1.3 of [DVB-GEM 1.1]. OCAP applications SHALL only include those security files defined in this profile. The DVB MHP security files SHALL NOT be used to authenticate the application but SHALL be listed in the appropriate ocap.hashfile for the directory in which they occur.

## 14.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section. Section 14 (this section) of the OCAP 1.1 Profile corresponds to Section 12 of [DVB-MHP1.1.2] and Section 12 of [DVB-GEM 1.1] as follows:

***Table 14–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 14 Security | 12 Security | Extension | 12 Security | Extension |
| 14.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 14.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 14.2.1 Deviations from the DVB-MHP Specification | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| No Corresponding Section | 12.1 Introduction | A subsection is extended | 12.1 Introduction | A subsection is extended |
| 14.2.1.1 Overview of the Security Framework for Applications | 12.1.1 Overview of the security framework for applications | Extension | 12.1.1 Overview of the security framework for applications | Extension |
| No Corresponding Section | 12.1.2 Overview of return channel security | Compliance | 12.1.2 Overview of return channel security | Compliance |
| No Corresponding Section | 12.1.3 Extensions to MHP application signing framework | Compliance | | |
| No Corresponding Section | 12.2 Authentication of applications | Compliance | 12.2 Authentication of applications | Compliance |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| No Corresponding Section | 12.3 Message transport | Compliance | 12.3 Message transport | Compliance |
| No Corresponding Section | 12.4 Detail of application authentication messages | A subsection is extended | 12.4 Detail of application authentication messages | A subsection is extended |
| No Corresponding Section | 12.4 Detail of application authentication messages | Compliance | 12.4.1 Hash File | A subsection is extended |
| No Corresponding Section | 12.4 Detail of application authentication messages | Compliance | 12.4.1.1 Description | Compliance |
| 14.2.1.2 HashFile Location and Naming Conventions | 12.4 Detail of application authentication messages | Compliance | 12.4.1.2 HashFile location and naming conventions | Extension |
| No Corresponding Section | 12.4 Detail of application authentication messages | Compliance | 12.4.1.3 Digest value computation rules | Compliance |
| No Corresponding Section | 12.4 Detail of application authentication messages | Compliance | 12.4.1.4 Warning concerning grouping of objects under a single digest (Informative) | Compliance |
| No Corresponding Section | 12.4 Detail of application authentication messages | Compliance | 12.4.1.5 Special authentication rules | Compliance |
| No Corresponding Section | 12.4 Detail of application authentication messages | Compliance | 12.4.2 Signature File | A subsection is extended |
| 14.2.1.3 SignatureFile Location and Naming Conventions | 12.4 Detail of application authentication messages | Compliance | 12.4.2.1 Description | Extension |
| 14.2.1.3 SignatureFile Location and Naming Conventions | 12.4 Detail of application authentication messages | Compliance | 12.4.2.2 SignatureFile location and naming conventions | Extension |
| No Corresponding Section | 12.4 Detail of application authentication messages | Compliance | 12.4.2.3 Supported algorithms | Compliance |
| No Corresponding Section | 12.4 Detail of application authentication messages | Compliance | 12.4.2.4 Signature computation rules | Compliance |
| No Corresponding Section | 12.4 Detail of application authentication messages | Compliance | 12.4.2.5 Authentication rules | Compliance |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| No Corresponding Section | 12.4 Detail of application authentication messages | Compliance | 12.4.3 Certificate File | A subsection is extended |
| No Corresponding Section | 12.4 Detail of application authentication messages | Compliance | 12.4.3.1 Description | Compliance |
| No Corresponding Section | 12.4 Detail of application authentication messages | Compliance | 12.4.3.2 ASN.1 encoding | Compliance |
| No Corresponding Section | 12.4 Detail of application authentication messages | Compliance | 12.4.3.3 Supported algorithms | Compliance |
| No Corresponding Section | 12.4 Detail of application authentication messages | Compliance | 12.4.3.4 Name matching | Compliance |
| 14.2.1.4 CertificateFile Location and Naming Conventions | 12.4 Detail of application authentication messages | Compliance | 12.4.3.5 CertificateFile location and naming conventions | Extension |
| No Corresponding Section | 12.4 Detail of application authentication messages | Compliance | 12.4.3.6 Authentication rules | Compliance |
| 14.2.1.15 Integration of the File Authentication Process | 12.4 Detail of application authentication messages | Compliance | 12.4.4 Integration | Extension |
| No Corresponding Section | 12.5 Profile of X.509 certificates for authentication of applications | Compliance | 12.5 Profile of X.509 certificates for authentication of applications | Extension |
| No Corresponding Section | 12.5 Profile of X.509 certificates for authentication of applications | Compliance | 12.5.1 signatureAlgorithm | Compliance |
| No Corresponding Section | 12.5 Profile of X.509 certificates for authentication of applications | Compliance | 12.5.2 signatureValue | Compliance |
| No Corresponding Section | 12.5 Profile of X.509 certificates for authentication of applications | Compliance | 12.5.3 version | Compliance |
| No Corresponding Section | 12.5 Profile of X.509 certificates for authentication of applications | Compliance | 12.5.4 issuer | Compliance |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| No Corresponding Section | 12.5 Profile of X.509 certificates for authentication of applications | Compliance | 12.5.5 validity | Compliance |
| No Corresponding Section | 12.5 Profile of X.509 certificates for authentication of applications | Compliance | 12.5.6 subject | Compliance |
| No Corresponding Section | 12.5 Profile of X.509 certificates for authentication of applications | Compliance | 12.5.8 Unique Identifiers | Compliance |
| 14.2.1.5 Profile of X.509 certifications for authentication of applications | 12.5 Profile of X.509 certificates for authentication of applications | Compliance | 12.5.9 Extensions | Extension |
| No Corresponding Section | 12.6 Security policy for appl35 | A subsection is extended | 12.6 Security policy for applications | A subsection is extended |
| 14.2.1.7 General Principles | 12.6 Security policy for appl35 | Extension | 12.6.1 General principles | Extension |
| 14.2.1.8 Permission Request File DTDs | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| No Corresponding Section | 12.6 Security policy for appl35 | Extension | 12.6.2 Permission request file | Compliance |
| No Corresponding Section | 12.6 Security policy for appl35 | Extension | 12.6.2.1 File encoding | Compliance |
| No Corresponding Section | 12.6 Security policy for appl35 | Extension | 12.6.2.1.1 Number representation | Compliance |
| No Corresponding Section | 12.6 Security policy for appl35 | Extension | 12.6.2.2 File integrity | Compliance |
| 14.2.1.9 Example | 12.6 Security policy for appl35 | Extension | 12.6.2.3 Example | Extension |
| 14.2.1.10 Permission Request File Name and Location | 12.6 Security policy for appl35 | Extension | 12.6.2.4 Permission request file name and location | Extension |
| No Corresponding Section | 12.6 Security policy for appl35 | Extension | 12.6.2.5 Permission request file | Compliance |
| No Corresponding Section | 12.6 Security policy for appl35 | Extension | 12.6.2.6 Credentials | Compliance |
| No Corresponding Section | 12.6 Security policy for appl35 | Extension | 12.6.2.7 File Access | Compliance |
| No Corresponding Section | 12.6 Security policy for appl35 | Extension | 12.6.2.8 CA API | Compliance |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| No Corresponding Section | 12.6 Security policy for appl35 | Extension | 12.6.2.9 Application lifecycle control policy | Compliance |
| No Corresponding Section | 12.6 Security policy for appl35 | Extension | 12.6.2.10 Return channel access policy | Compliance |
| No Corresponding Section | 12.6 Security policy for appl35 | Extension | 12.6.2.11 Tuning access policy | Compliance |
| No Corresponding Section | 12.6 Security policy for appl35 | Extension | 12.6.2.12 Service selection policy | Compliance |
| No Corresponding Section | 12.6 Security policy for appl35 | Extension | 12.6.2.13 Media API access policy | Compliance |
| No Corresponding Section | 12.6 Security policy for appl35 | Extension | 12.6.2.14 Inter-application communication policy | Compliance |
| No Corresponding Section | 12.6 Security policy for appl35 | Extension | 12.6.2.15 User Setting and Preferences access policy | Compliance |
| No Corresponding Section | 12.6 Security policy for appl35 | Extension | 12.6.2.16 Network permissions | Compliance |
| No Corresponding Section | 12.6 Security policy for appl35 | Extension | 12.6.2.17 Dripfeed permissions | Compliance |
| 14.2.1.11 Scenario Example | 12.7 Example of creating an application that can be authenticated | Compliance | 12.7 Example of creating an application that can be authenticated | Extension |
| 14.2.1.12 OCAP 1.1 Certification Procedures | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| No Corresponding Section | 12.8 GEM/MHP certification procedures | Compliance | 12.8 MHP certification procedures | Compliance |
| 14.2.1.13 Signature and Certificate Chain Validation | 12.9 Certificate management | Extension | 12.9 Certificate management | Extension |
| No Corresponding Section | 12.10 Security on the return channel | Compliance | 12.10 Security on the return channel | Compliance |
| No Corresponding Section | 12.11 The internet profile of X.509 (informative) | A subsection is extended | 12.11 The internet profile of X.509 (informative) | A subsection is extended |
| No Corresponding Section | 12.11 The internet profile of X.509 (informative) | Compliance | 12.11.2 Standard certificate extensions | A subsection is extended |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 14.2.1.14 SignatureFile Description | 12.11 The internet profile of X.509 (informative) | Compliance | 12.11.2.1 Authority key identifier | Extension |
| No Corresponding Section | 12.11 The internet profile of X.509 (informative) | Compliance | 12.11.2.2 Subject key identifier | Compliance |
| No Corresponding Section | 12.11 The internet profile of X.509 (informative) | Compliance | 12.11.2.3 Key usage | Compliance |
| No Corresponding Section | 12.11 The internet profile of X.509 (informative) | Compliance | 12.11.2.4 Private key usage period | Compliance |
| No Corresponding Section | 12.11 The internet profile of X.509 (informative) | Compliance | 12.11.2.5 Certificate policies | Compliance |
| No Corresponding Section | 12.11 The internet profile of X.509 (informative) | Compliance | 12.11.2.6 Policy mappings | Compliance |
| No Corresponding Section | 12.11 The internet profile of X.509 (informative) | Compliance | 12.11.2.7 Subject Alternative Name | Compliance |
| No Corresponding Section | 12.11 The internet profile of X.509 (informative) | Compliance | 12.11.2.8 Issuer Alternative Name | Compliance |
| No Corresponding Section | 12.11 The internet profile of X.509 (informative) | Compliance | 12.11.2.9 Subject Directory attributes | Compliance |
| No Corresponding Section | 12.11 The internet profile of X.509 (informative) | Compliance | 12.11.2.10 Basic Constraints | Compliance |
| No Corresponding Section | 12.11 The internet profile of X.509 (informative) | Compliance | 12.11.2.11 Name Constraints | Compliance |
| No Corresponding Section | 12.11 The internet profile of X.509 (informative) | Compliance | 12.11.2.12 Policy Constraints | Compliance |
| No Corresponding Section | 12.11 The internet profile of X.509 (informative) | Compliance | 12.11.2.13 Extended key usage field | Compliance |
| No Corresponding Section | 12.11 The internet profile of X.509 (informative) | Compliance | 12.11.2.14 CRL Distribution points | Compliance |
| No Corresponding Section | 12.12 Platform minima | Compliance | 12.12 Platform minima | Compliance |
| No Corresponding Section | 12.13 Plug-ins | Compliance | 12.13 Plug-ins | Compliance |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| No Corresponding Section | 12.14 Applications loaded from an interaction channel | Compliance | 12.14 Applications loaded from an interaction channel | Compliance |
| No Corresponding Section | 12.15 Stored applications | Compliance | 12.15 Stored applications | Compliance |
| 14.2.2 Extensions to DVB-GEM (Normative) | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |

## 14.2 OCAP 1.1 Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

### 14.2.1 Deviations from the DVB-MHP Specification

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

#### *14.2.1.1 Overview of the Security Framework for Applications*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 12.1.1 Overview of the security framework for applications and are extensions of [DVB-MHP1.1.2] Section: 12.1.1 Overview of the security framework for applications.

Section 12.1.1 of [DVB-GEM 1.1] applies to EE applications executing on OCAP 1.1. The phrase application code, as used in that section, applies to Java byte codes. Signed applications are to be interpreted as applications that are trusted though the mechanisms defined in this section of OCAP 1.1.

#### *14.2.1.2 HashFile Location and Naming Conventions*

This subsection is compliant with [DVB-GEM 1.1] Section: 12.4 Detail of application authentication messages and are extensions of [DVB-MHP1.1.2] Section: 12.4.1.2 HashFile location and naming conventions.

This section of the OCAP 1.1 Profile extends Section 12.4 of [DVB-GEM 1.1], which references Section 12.4.1.2 of [DVB-MHP1.1.2]. OCAP defines a new hash file in addition to the hash file defined by DVB.

The name of the hash file, used for OCAP applications, which is placed in each directory that contains files that need to be authenticated, SHALL be called `ocap.hashfile`. The use of ocap.hashfile is described in Section 14.2.1.11.

#### *14.2.1.3 SignatureFile Location and Naming Conventions*

This subsection is compliant with [DVB-GEM 1.1] Section: 12.4 Detail of application authentication messages and contains extensions of [DVB-MHP1.1.2] Section: 12.4.2.1 Description and also contains requirements that are extensions to [DVB-MHP1.1.2] Section: 12.4.2.2 SignatureFile location and naming conventions.

This section extends Section 12.4 of [DVB-GEM 1.1] and the corresponding Section 12.4.2.1 of [DVB-MHP1.1.2]. and defines a new signature file naming convention.

By convention, the name of the signature file, used for OCAP applications, SHALL be 'ocap.signaturefile.'<x>, where <x> is a string that distinguishes between several possible signature files. Other than this change to the naming convention, this OCAP 1.1 Profile complies with Section 12.4.2.2 of [DVB-MHP1.1.2] corresponding to Section 12.4 of [DVB-GEM 1.1].

#### *14.2.1.4 CertificateFile Location and Naming Conventions*

This subsection is compliant with [DVB-GEM 1.1] Section: 12.4 Detail of application authentication messages and contains extensions of [DVB-MHP1.1.2] Section: 12.4.3.5 CertificateFile location and naming conventions.

The name of a certificate file SHALL be 'ocap.certificates.'<x>, where <x> is identical to the extension of the signature file name that is authenticated by the OCAP certificate chain in this file. Other than the name of this file defined here, the OCAP certificate requirements are defined in Section 6 of [OC-SEC].

#### *14.2.1.5 Profile of X.509 certifications for authentication of applications*

This subsection is compliant with [DVB-GEM 1.1] Section: 12.5 Profile of X.509 certificates for authentication of applications and contains extensions of [DVB-MHP1.1.2] Section: 12.5.9 Extensions.

The referenced section of [DVB-GEM 1.1] applies only to broadcast MHP applications, so far as this profile is concerned, it SHALL be considered as applicable to all OCAP applications. Section 6 in [OC-SEC] SHALL be considered as applicable for all OCAP applications including the Monitor Application. The certificates that contain the key used to sign Monitor Applications also follow the format described in Section 6 of [OC-SEC] as do certificates which are referred to by persistentfilecredentials.

**Note:** The value of the organisation_id contained in the subject field of a "distinguished name" found in a certificate used to authenticate a shared class library need not match the organisation_id of an application that makes reference to the shared class library. See Section 21.2.1.21 for further information on share class libraries.

#### *14.2.1.6 subjectPublicKeyInfo*

This subsection is compliant with [DVB-GEM 1.1] Section: 12.5 Profile of X.509 certificates for authentication of applications and contains extensions of [DVB-MHP1.1.2] Section: 12.5.7 SubjectPublic Key Info.

The key lengths that implementations are required to support are documented in [OC-SEC].

#### *14.2.1.7 General Principles*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 12.6 Security policy for applications and contains extensions of [DVB-MHP1.1.2] Section: 12.6.1 General principles.

To expand on that section, when file version information is received which corresponds to incorrectly signed or missing data, the OCAP 1.1 terminal can continue to use an older version.

#### *14.2.1.8 Permission Request File DTDs*

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

An OCAP receiver SHALL support both the [DVB-MHP1.1.2] Permission Request File (PRF) DTD and an extended PRF DTD called the OCAP Permission Request File DTD, defined in Section 14.2.2.1, as allowed by [DVB-GEM 1.1].

If both an MHP PRF and an OCAP PRF are in the same directory as the initial file of the OCAP application, the MHP PRF shall be ignored.

The return channel element and, in particular, the phone number element defined in [DVB-GEM 1.1], may not have corresponding semantics in an OCAP environment. For clarification, an OCAP 1.1 implementation is required to interpret the presence of the phone number element. It is, however, not required to process it.

#### *14.2.1.9 Example*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 12.6 Security policy for appl35 and contains extensions of [DVB-MHP1.1.2] Section: 12.6.2.3 Example

Because OCAP 1.1 has redefined several elements of the Permission Request File DTD, a different example from that used in Section 12.6.2.3 of [DVB-MHP1.1.2], referred by 12.6 of [DVB-GEM 1.1] is necessary:

```
<?xml version="1.0" standalone="yes" ?>

<!DOCTYPE permissionrequestfile PUBLIC "-//OCAP//DTD Permission Request File 1.0//EN"
"http://www.opencable.com/ocap/dtd/ocappermissionrequestfile-1-1.dtd">

<!-- see MHP 1.1.2 section 12.6.2.1 and OCAP 1.1 section 14.2.1.8 for details-->

<permissionrequestfile orgid="0xFFFF" appid="0x0001" xmlns:ocap="ocap">

    <file value="true"></file>
    <applifecyclecontrol value="true"></applifecyclecontrol>
    <returnchannel>
        <defaultisp></defaultisp> <!-- default ISP connection -->
        <phonenumber></phonenumber>   <!-- any phone number -->
    </returnchannel>
    <tuning value="true"></tuning>
    <servicesel value="true"></servicesel>
    <userpreferences read="true" write="true"></userpreferences>
    <network>
        <host action="all">nethostname</host>
    </network>
    <dripfeed value="true"></dripfeed>

<!-- Authentication information see MHP 1.1.2 section 12.6.2.6-->
    <persistentfilecredential>
        <grantoridentifier id="0xFF"></grantoridentifier>
        <expirationdate date="4/27/2005"></expirationdate>
        <filename read="true" write="true">foo.bar</filename>
        <filename read="true" write="true">foo/foobar.bar</filename>
        <signature>123456</signature>
        <certchainfileid>3</certchainfileid>
    </persistentfilecredential>

<!-- various Permissions, see OCAP 1.1 profile for details-->

    <ocap:monitorapplication name="registrar" value="true"></ocap:monitorapplication>
    <ocap:monitorapplication name="service" value="true"></ocap:monitorapplication>
<!-- etc -->

    <ocap:servicetypepermission type="broadcast" action="all"
value="true"></ocap:servicetypepermission>
    <ocap:servicetypepermission type="abstract.mso" action="all"
value="true"></ocap:servicetypepermission>
<!-- etc -->

</permissionrequestfile>
```

#### *14.2.1.10 Permission Request File Name and Location*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 12.6 Security policy for applications and contains extensions of [DVB-MHP1.1.2] Section: 12.6.2.4 Permission request file name and location.

Except for the name of the permission request file, which SHALL be ocap.<application name>.perm, and an addition to Table 54, which lists the value 0x0003 for OCAP, this profile complies with Section 12.6 of [DVB-MHP1.1.2].

#### *14.2.1.11 Scenario Example*

This subsection is based on [DVB-GEM 1.1] Section: 12.7 Example of creating an application that can be authenticated and contains extensions of [DVB-MHP1.1.2] Section: 12.7 Example of creating an application that can be authenticated.

The example shown in Figure 15 of Section 12.7.1 of [DVB-MHP1.1.2], referenced by 12.7 of [DVB-GEM 1.1] except the DVB example, which shows a file system carrying only a single application, would be replaced with one in which the names of the corresponding hash files would be ocap.hashfile rather than dvb.hashfile.

Additionally, the corresponding Table 57 for the OCAP 1.1 example would be as shown in Table 14–2 below:

***Table 14–2 - root/Xlet1/classes/subclasses/ocap.hashfile***

| Field | Comment |
|---|---|
| 2 | Two digests |
| 1 | Type of digest algorithm = MD5 |
| 1 | Number of entries over which an MD5 hash has been computed |
| Sub1.class | First entry |
| H1 | MD5 hash of file sub1.class |
| 1 | Type of digest algorithm = MD5 |
| 1 | Number of entries over which an MD5 hash has been computed |
| Sub2.class | Second entry |
| H2 | MD5 hash of file sub2.class |

In addition, there are two changes in Section 12.7.2 of [DVB-GEM 1.1] in order to correlate with this example and comply with OCAP 1.1:

- Change of Table 58, in which <ocap.Xlet1.perm> file is included;
- Change of Table 59, in which <ocap.certificates.1> files is included.

#### *14.2.1.12 OCAP 1.1 Certification Procedures*

This subsection contains OCAP-specific requirements that does not correspond to any [DVB-GEM 1.1] Section.

This OCAP 1.1 certification procedure describes the administration of certificates, see Section 6 of [OC-SEC].

Each leaf certificate SHALL contain the `organization_id` in the organization name attribute. In case there are several root certificate entities, the certificate procedure SHALL ensure that the leaf certificate contain the `organization_id` defined by DVB.

#### *14.2.1.13 Signature and Certificate Chain Validation*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 12.9 Certificate management and contains extensions of [DVB-MHP1.1.2] Section: 12.9 Certificate management.

This section extends Section 12.9 of [DVB-GEM 1.1] and corresponding Section 12.9.2 of [DVB-MHP1.1.2].

To correctly authenticate a subtree there SHALL be a valid "chain" of certificates from the signature to a root certificate as is illustrated in Figure 14–1. In OCAP 1.1 the AuthorityKeyIdentifier extension in a certificate SHALL match the SubjectKeyIdentifier in the issuer's certificate in order for a valid chain of certificates to be established.

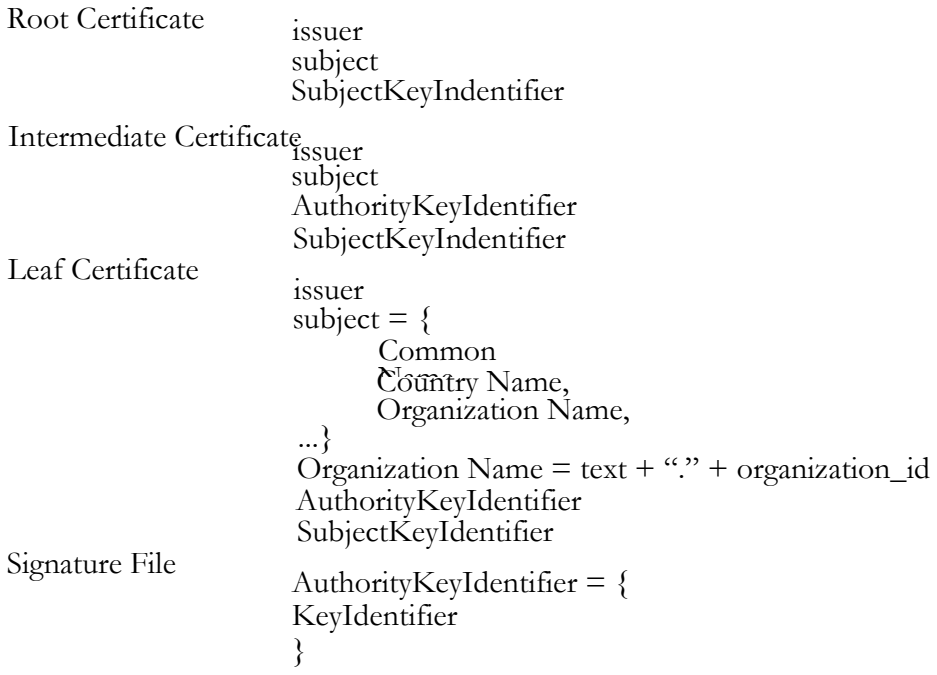



***Figure 14–1 - Certificate Chain Illustrated***

#### *14.2.1.14 SignatureFile Description*

This subsection is compliant with [DVB-GEM 1.1] Section: 12.11 The internet profile of X.509 (informative) and contains extensions of [DVB-MHP1.1.2] Section: 12.11.2.1 Authority key identifier.

The OCAP 1.1 Profile extends section 12.1 of [DVB-GEM 1.1] and the corresponding Section 12.1.2.1 of [DVB-MHP1.1.2] by changing the required content of the AuthorityKeyIdentifier sequence as follows:

The AuthorityKeyIdentifier structure SHALL contain the keyIdentifier element and this SHALL be used to validate the signature file to the leaf certificate by matching this value against the corresponding value in the leaf certificate's SubjectKeyIdentifier. The implementation is not required to provide the authorityCertIssuer or the authorityCertSerialNumber elements of the AuthorityKeyIdentifier.

#### *14.2.1.15 Integration of the File Authentication Process*

The OCAP 1.1 Profile modifies the [DVB-GEM 1.1] file authentication process in Section 12.4 and the corresponding Section 12.4.4 of [DVB-MHP1.1.2], in order to ensure that an application that is delivered with two or more signature files each with a corresponding certificate file that contains a common root certificate is authenticated through all these signature files.

Logically a file is authenticated as follows:

1. Confirm that the file is listed in the hash file located in the same directory as the file to be authenticated.
2. Verify that the file contents and the corresponding digest value are consistent.

3. Recursively ascend the directory hierarchy checking that each directory is authenticated by its parent directory until a directory is found that contains one or more signature files. Such a directory is termed the root directory of an authenticated subtree.

**Note:** This means that there is no requirement to progress above the root of the authenticated subtree to examine further hash files. If such a directory has a digest type other than "Non authenticated" in its parent directory, this is only significant when verifying the hash file of the parent directory.

**Note:** The presence of more than one signature file enables more than one set of organizations to authenticate a subtree.

4. For a signature file locate the corresponding certificate file (where the x portion of the signature file's file name identifies the certificate file to be used).
5. If the certificate file does not contain a root certificate that matches one of the root certificates installed in the OCAP terminal, return to step (3) to locate any further signature files.

**Note:** OCAP 1.1 requires that each certificate file contains a root certificate.

6. Use the corresponding certificate file to verify that the signature correctly signs the hash file.
7. Follow the certificate chain contained within the certificate file verifying each link in the chain until the link to the root certificate is found.
8. If the identified root certificate and all the intermediate certificates leading to it are "satisfactory", accept the files as being authenticated for this signature file. "satisfactory" depends on the policies implemented in the receiver and other constraints expressed in this profile and in [OC-SEC]. In particular the requirements in Section 12.2 of [DVB-GEM 1.1] which points to [DVB-MHP1.1.2] 11.2.3, "Class Loading" for using the "same" leaf certificate to authenticate OCAP-J class files SHALL be observed.
9. If the application is an unbound application that requires multiple signatures, return to step (4) and repeat for the other signature files until the following conditions are met:
    - The application is authenticated by at least two signature files.
    - The certificate files used to authenticate these two signature files contain a common root certificate.
    - The SHA-1 hash of the leaf certificates in one of these certificate files matches one of the SHA-1 hashes included in the privileged certificate descriptor.

**Note:** This step in the authentication process applies to applications. If an application needs to verify that a DSMCC object has been signed by a privileged certificate, it should call org.dvb.dsmcc.DSMCCObject.getSigners(known_root) with known_root set to true and then call org.ocap.application.isPrivilegedCertificate(Certificate) for each of the end-entity certificates in the array returned.

10. Dependant on receiver policy return to step (4) and repeat for other signature files.

**Note:** The above is a logical description of the process and does not constrain implementations to perform these steps in this exact order (e.g., hash files may be verified when descending a directory hierarchy rather than ascending one).

A file system may contain several independent authenticated sub-trees, each tree with its own subtree root directory.

### 14.2.2 Extensions to DVB-GEM (Normative)

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

The following information is normative for the OCAP 1.1 Profile. It is specific to OCAP 1.1 and is not contained within the [DVB-GEM 1.1].

#### *14.2.2.1 OCAP Permission Request File*

##### *14.2.2.1.1 DTD definition*

The OCAP Permission Request File (PRF) DTD is extending the [DVB-MHP1.1.2] Permission Request File DTD since additional permissions have been added to meet OCAP 1.1 requirements. However, in order to be compliant with the [DVB-GEM 1.1], the OCAP 1.1 PRF DTD includes all elements and attributes of the GEM 1.0 PRF DTD. The OCAP Permission Request File is an XML File.

The following formal public identifier SHALL be used to identify the OCAP PRF DTD:

```
"-//OCAP//DTD Permission Request File 1.0//EN"
```

and the following URL for the SystemLiteral may be used to reference this file:

```
http://www.opencable.com/ocap/dtd/ocappermissionrequestfile-1-0.dtd
```

The Name used in the document type declaration shall be "permissionrequestfile".

The OCAP PRF DTD is provided below:

```
<!-- ..................................................................... -->
<!-- OCAP 1.1 Permission Request File DTD -->
<!-- ..................................................................... -->
<!-- file: ocappermissionrequestfile-1-0.dtd -->
<!-- ..................................................................... -->
<!-- This is the DTD for the OCAP 1.1 permission request file.
The following formal public identifier SHALL be used to identify it:
"-//OCAP//DTD Permission Request File 1.0//EN"
The following URL for the SystemLiteral may be used to reference this file :
http://www.opencable.com/ocap/dtd/ocappermissionrequestfile-1-0.dtd
-->
<!-- All elements and attributes defined in MHP 1.0.x shall be supported and -->
<!-- inserted at this level with appropriate extensions. OCAP extends the -->
<!-- set of permissions with the ocap:monitorapplication one. -->
<!ELEMENT permissionrequestfile (file?, capermission?, applifecyclecontrol?, returnchannel?,
tuning?, servicesel?, userpreferences?, network?, dripfeed?, persistentfilecredential*,
ocap:monitorapplication*, ocap:servicetypepermission*, ocap:registeredapi.user*)>
<!ATTLIST permissionrequestfile
   orgid CDATA #REQUIRED
   appid CDATA #REQUIRED
   xmlns:ocap CDATA #FIXED
>
<!ELEMENT file EMPTY>
<!ATTLIST file
   value (true | false) "true"
>
<!ELEMENT capermission (casystemid)+>
<!ELEMENT casystemid EMPTY>
<!ATTLIST casystemid
   entitlementquery (true | false) "false"
   id CDATA #REQUIRED
   mmi (true | false) "false"
   messagepassing (true | false) "false"
   buy (true | false) "false"
>
<!ELEMENT applifecyclecontrol EMPTY>
<!ATTLIST applifecyclecontrol
   value (true | false) "true"
>
<!ELEMENT returnchannel (defaultisp?, phonenumber*)>
<!ELEMENT defaultisp EMPTY>
<!ELEMENT phonenumber (#PCDATA)>
<!ELEMENT tuning EMPTY>
<!ATTLIST tuning
   value (true | false) "true"
>
<!ELEMENT servicesel EMPTY>
<!ATTLIST servicesel
   value (true | false) "true"
>
<!ELEMENT userpreferences EMPTY>
<!ATTLIST userpreferences
   write (true | false) "false"
   read (true | false) "true"
```

```
>
<!ELEMENT network (host)+>
<!ELEMENT host (#PCDATA)>
<!ATTLIST host
   action CDATA #REQUIRED
>
<!ELEMENT dripfeed EMPTY>
<!ATTLIST dripfeed
   value (true | false) "true"
>
<!ELEMENT persistentfilecredential (grantoridentifier, expirationdate, filename+, signature,
certchainfileid)>
<!ELEMENT grantoridentifier EMPTY>
<!ATTLIST grantoridentifier
   id CDATA #REQUIRED
>
<!ELEMENT expirationdate EMPTY>
<!ATTLIST expirationdate
   date CDATA #REQUIRED
>
<!ELEMENT filename (#PCDATA)>
<!ATTLIST filename
   write (true | false) "true"
   read (true | false) "true"
>
<!ELEMENT signature (#PCDATA)>
<!ELEMENT certchainfileid (#PCDATA)>
<!-- In addition, the following elements and attributes are defined in order
to support OCAP specific behavior. They are prefixed by the string "ocap:" according to DVB-GEM
-->
<!ELEMENT ocap:servicetypepermission EMPTY>
<!ATTLIST ocap:servicetypepermission
   type (broadcast | abstract.mso | abstract.manufacturer) "broadcast"
   action (own | all) "all"
   value (true | false) "false"
>
<!ELEMENT ocap:monitorapplication EMPTY>
<!ATTLIST ocap:monitorapplication
   value (true | false) "false"
   name (registrar | handler.registrar | service | servicemanager | security | reboot |
systemevent | handler.appFilter | handler.resource | handler.closedCaptioning |
handler.analogAudioControl | handler.registrar | filterUserEvents | handler.eas | setVideoPort |
podApplication | signal.configured | properties | storage | registeredapi | registeredapi.manager
| vbifiltering | codeDownload | mediaAccess | quarantine | environment.selection |
testApplications | diagnostics)
#IMPLIED
>
<!ELEMENT ocap:registeredapi.user EMPTY>
<!ATTLIST ocap:registeredapi.user
name CDATA REQUIRED
>
```

#### *14.2.2.1.2 OCAP Permission Request File name and location*

The format for the OCAP Permission Request File name shall be: 'ocap.'<application name>'.perm'

The prefix "ocap" identifies this as a well known file specified by this profile. The portion "application name" carries the file name of the initial file of the application (see Section 14.2.1.10).

The OCAP permission request file shall be located in the same directory as the initial file.

### *14.2.2.2 Monitor Application Permission*

Monitor Application Permission can provide a set of permissions typically required by a Monitor Application. The network MAY signal these permissions for an application using the permission request file defined in Section 14.2.2.1. Multiple instances of the "ocap:monitorapplication" element may appear, one for each type of permission that is requested. For a detailed discussion of each Monitor Application permission, refer to Annex Q. The following policy is applied to Monitor Application Permissions:

#### *14.2.2.2.1 Unsigned applications*

An unsigned application may not use any Monitor Application Features.

#### *14.2.2.2.2 Signed applications*

By default, a signed application may not use any Monitor Application capabilities. However, the right to exercise specific Monitor Application capabilities can be requested with the Monitor Application Permission that can be put in the Permission Request File. All applications with any Monitor Application permission SHALL be authenticated by both the application author's certificate and by either the MSO application verification certificate or the OpenCable application verification certificate. If these conditions are not met then Monitor Application permissions SHALL NOT be granted although the application SHALL NOT be prevented from running. Only Monitor Application permissions SHALL be denied if the authentication rules described here are not met.

### *14.2.2.3 Service Type Permission*

By default a signed application is granted the ServiceTypePermission specified in Section 10.2.2.2.3 as appropriate to its application type.

When a ServiceTypePermission is contained within an application's permission request file the permission is granted if the ocap:value attribute is true, otherwise it is revoked. The ocap:type and ocap:actions attributes determine the sets of service types and service contexts affected by the permission entry respectively.

### *14.2.2.4 Privileged Monitor Application API Access*

All applications that use any of the Monitor Application privileges SHALL be signed by both the application author and either the MSO or CableLabs. The permission file SHALL be included in the set of files that are authenticated by these dual signatures. This ensures that the MSO has agreed to allow this application to access these privileged APIs.

### *14.2.2.5 CRL File Location and Naming Convention*

For CRLs that are authenticated by a broadcast certificate that uses the OCAP X.509 profile, the format of the name of files carrying CRLs SHALL be: 'ocap.crl.'<x>. In this case the <x> portion of the file name corresponds to the <x> portion of a certificate file name for the certificate file that authenticates the CRL.

For CRLs that are authenticated by an OpenCable root certificate the format of the name of files carrying CRLs SHALL be: 'ocap.crl.root.'<x>. In this case, the <x> portion of the file name is just a discriminator to ensure non-collision of CRL file names in the event that there is more than one in this directory.

The CRL file name MAY not be constant through time or across broadcasts. So, implementations SHALL NOT rely on this file name when caching the CRL.

All CRL files SHALL be located in a subdirectory of the root of the file system called ocap.crl. The location of certificate files authenticating the CRL files SHALL follow the same rules as for the location of certificates relative to a signature file. That is, the certificate files SHALL either be in the ocap.crl directory or in the root directory.

### *14.2.2.6 Examples*

This OCAP 1.1 Profile differs from the [DVB-GEM 1.1], Section 12.9.1.6.1 as follows. The CRL file MAY be obtained in another way, as noted above, rather than by selecting one of broadcaster A's channels.

### *14.2.2.7 Root Certificate Management*

This section extends Section 12.9.2 of [DVB-GEM 1.1].

In OCAP 1.1 Root Certificate Management uses the common download facility for downloading code images and certificate information as described in [OC-SEC].

The Root Certificate Management Messages described in [DVB-GEM 1.1] are not used in OCAP 1.1 to replace OpenCable root certificates.

#### *14.2.2.8 Persistent Storage Access*

The implementation's assessment of Persistent Storage access rights SHALL follow the steps below in sequence and SHALL use the access rights granted (or denied) at the step where a match of application_id / organization_id is found. The following SHALL apply to Logical Storage Volume (LSV), directories and files in persistent storage.

a. If the application_id and organization_id of the accessing application match the corresponding fields in the LSV, directory, or file Extended File Access Permission, then the readApplicationAccessRight and writeApplicationAccessRight determine access to the LSV, directory or file.

b. If the application_id of the accessing application does not match the corresponding field in the LSV, directory or file, but the organization_id of the accessing application does match the corresponding field in the LSV, directory or file Extended File Access Permission, then the readOrganizationAccessRight and writeOrganizationAccessRight determine access to the LSV, directory or file.

c. If the organization_id of the accessing application does not match the corresponding field in the LSV, directory or file, but the organization_id of the accessing application does match one of the other organization_id fields in the LSV, directory or file Extended File Access Permission, then the OtherOrganizationsReadAccessRights and OtherOrganizationsWriteAccessRights determine access to the LSV, directory or file.

d. If the organization does not match the organization_id or one of the other organization_id fields in the LSV, directory or file, then the readWorldAccessRight and writeWorldAccessRight determine access to the LSV, directory or file.

# 15 GRAPHICS REFERENCE MODEL

This section provides a reference model for controlling and managing video, interface elements such as GUI widgets, and raw graphical primitives such as points and lines. The [DVB-MHP1.1.2] model expects three planes:

- background plane
- video plane
- graphics plane

OCAP 1.1 follows this reference model. Using this reference model, an OCAP 1.1 application has access to all three planes and MAY use them to do the following:

- place video, GUI widgets, and graphics inside a contiguous rectangular region of the graphics plane
- control video on the video plane, outside the GUI widget hierarchy
- place still images or solid color in the background plane

The Graphics Reference Model defines a number of coordinate systems used for different purposes and includes a way to transform between them as needed.

Finally, this section provides a reference model showing how the presentation of closed-captioning is controlled.

## 15.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that does not correspond to any DVB-GEM 1.2.1 Section. Section 15 (this section) corresponds to Section 13 of [DVB-MHP1.1.2] and [DVB-GEM 1.1] as follows:

***Table 15–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 15 Graphics Reference Model | 13 Graphics reference model | Extension | 13 Graphics reference model | Extension |
| 15.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 15.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 15.2.1 Deviations from the DVB-MHP Specification | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 15.2.1.1 Introduction | 13.0 General | Extension | 13.1 Introduction | Extension |
| No Corresponding Section | 13.1 Supported graphics resolution | Compliance | No Corresponding Section | |
| No Corresponding Section | 13.1 Supported graphics resolution | Compliance | 13.1.1 Interapplication interaction | |
| No Corresponding Section | 13.1 Supported graphics resolution | Compliance | 13.2 General Issues | A subsection is extended |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| No Corresponding Section | 13.1 Supported graphics resolution | Compliance | 13.2.1 Coordinate Spaces | A subsection is extended |
| No Corresponding Section | 13.1 Supported graphics resolution | Compliance | 13.2.1.1 Normalized screen space | Compliance |
| No Corresponding Section | 13.1 Supported graphics resolution | Compliance | 13.2.1.2 User space | Compliance |
| 15.2.1.2 Typical Resolutions | 13.1 Supported graphics resolution | Compliance | 13.2.1.3 Pixel Aspect Ratio | Extension |
| No Corresponding Section | 13.1 Supported graphics resolution | Compliance | 13.2.1.4 Video space | Compliance |
| No Corresponding Section | 13.1 Supported graphics resolution | Compliance | 13.3 Graphics | A subsection is extended |
| No Corresponding Section | 13.1 Supported graphics resolution | Compliance | 13.3.1 Modeling of the MHP display stack composition | Compliance |
| No Corresponding Section | 13.1 Supported graphics resolution | Compliance | 13.3.2 AWT Reference Model in the MHP | Compliance |
| No Corresponding Section | 13.1 Supported graphics resolution | Compliance | 13.3.3 HAVi devices and AWT components | Compliance |
| 15.2.1.3 Composition of AWT Containers and Components | 13.1 Supported graphics resolution | Compliance | 13.3.4 Composition | Extension |
| 15.2.1.3.1 Mixing of Native Components and Lightweight Components (Informative) | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| No Corresponding Section | 13.2 Aspect Ratio | Extension | 13.3.7 14:9 Aspect Ratio Support | Compliance |
| No Corresponding Section | 13.3 Broadcast streaming formats | Compliance | 13.4.1 Component based players and background players | Compliance |
| 15.2.1.4 Modeling MPEG Decoding and Presentation Pipeline | 13. Graphics Reference Model | Extension | 13.4.2 Modeling MPEG decoding and presentation pipeline | Extension |
| 15.2.1.5 Coordinate Spaces for Video | 13. Graphics Reference Model | Extension | 13.4.3 Coordinate Spaces | Extension |
| No Corresponding Section | 13. Graphics Reference Model | Compliance | 13.4.4 Video components | Compliance |
| No Corresponding Section | 13.4 Subtitles | Compliance | 13.5 Subtitles | Compliance |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| No Corresponding Section | No Corresponding Section | | 13.6 Approximations | Compliance |
| 15.2.2 Extensions to DVB-GEM (Normative) | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |

## 15.2 OCAP 1.1 Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

### 15.2.1 Deviations from the DVB-MHP Specification

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

#### *15.2.1.1 Introduction*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 13.0 General and contains extensions of [DVB-MHP1.1.2] Section: 13.1 Introduction.

In the OCAP 1.1 graphics reference model the background plane is a logical construct, and may not necessarily be implemented in hardware.

OCAP 1.1 does not support DVB subtitles. An OCAP 1.1 implementation is not required to provide the DVB-MHP subtitle API or behavior discussed in this section of the [DVB-MHP1.1.2].

#### *15.2.1.2 Typical Resolutions*

This subsection is compliant with [DVB-GEM 1.1] Section: 13.1 Supported graphics resolution and contains extensions of [DVB-MHP1.1.2] Section: 13.2.1.3 Pixel Aspect Ratio.

Typical resolutions for a full-screen graphics device, HGraphicsDevice, are shown in Table 15–2. This Table replaces Table 62, Typical Resolutions and their pixel aspect ratio (informative) found in Section 13.2.1.3 of the [DVB-MHP1.1.2]. The supported resolutions for OCAP 1.1 are defined in Section 25.

***Table 15–2 - Typical Resolutions and their Pixel Aspect Ratio***

| Resolutions for full-screen HGraphicsDevice | 4:3 Display Pixel Aspect Ratio | 16:9 Display Pixel Aspect Ratio |
|---|---|---|
| 640 x 480 | 1:1 | 4:3 |
| 960x540 | 3:4 | 1:1 |

#### *15.2.1.3 Composition of AWT Containers and Components*

This subsection is compliant with [DVB-GEM 1.1] Section: 13.1 Supported graphics resolution and contains extensions of [DVB-MHP1.1.2] Section: 13.3.4 Composition.

This section is an informative description for Section 13.0 of the [DVB-GEM 1.1].

If the AWT components are implemented as the heavyweight components, the discussion in Section 13.3 of [DVB-MHP1.1.2] is also applied to the AWT native or peer hierarchy. The normal AWT paint rules SHALL be followed for both the HAVi and native component hierarchies.

#### *15.2.1.3.1 Mixing of Native Components and Lightweight Components (Informative)*

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

The following information is informative to the OCAP 1.1 implementation.

In DVB-MHP and HAVi, components are implemented as lightweight components; there are no heavyweight/peer components. The HScene acts like a heavyweight component in that repaint commands bubble up through the HAVi container hierarchy until it reaches the root container (`HScene`). Then, the `paint()` method is called back down through the container stack to update the display.

In the OCAP Sun Java Specification [PBP1.1], there are native (heavyweight, use of peer) and lightweight components. An application can freely mix lightweight and native components in lightweight or native containers.

Mixing of HAVi components and [PBP1.1] peer components is problematic for OCAP 1.1 as well as application developers. To minimize the issues that arise from mixing native and lightweight components, the following guidelines SHOULD be used:

- Heavyweight components can be accommodated in HScenes as long as the HScene is implemented as a heavyweight itself. Indeed, the HAVi specification requires that HScene "act" like a heavyweight in some instances. Sun Microsystems has found it useful to implement HScene as a heavyweight `java.awt.Panel`. The additional plumbing required to support heavyweights in HScenes can be done without violation to the [HAVi] or [PBP1.1] specifications.
- The primary problem with mixing heavyweight and lightweight components in the same container involves confusion of z-ordering. This is a problem that exists apart from HAVi/AWT integration and has been around since JDK 1.1 added support for lightweight components. Instead of normatively prohibiting such mixing, this profile requires that OCAP 1.1 implementations SHALL be aware of this problem and address it using industry standard solutions.

### *15.2.1.4 Modeling MPEG Decoding and Presentation Pipeline*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 13. Graphics Reference Model and contains extensions of [DVB-MHP1.1.2] Section: 13.4.2 Modeling MPEG decoding and presentation pipeline.

References to DVB ETR 154 Standard Definition SHALL be replaced with references to [HOST 2.0].

### *15.2.1.5 Coordinate Spaces for Video*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 13. Graphics Reference Model and contains extensions of [DVB-MHP1.1.2] Section: 13.4.3 Coordinate Spaces.

This section is compliant with Section 13.0 of the [DVB-GEM 1.1] except for the following modifications: References to [ETSI TR 101 154] Standard Definition SHALL be replaced with references to [HOST 2.0].

## 15.2.2 Extensions to DVB-GEM (Normative)

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

The following information is normative for the OCAP 1.1 Profile. It is specific to OCAP 1.1 and is not contained within the [DVB-MHP1.1.2].

#### *15.2.2.1 Closed-Captioning*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 13 Graphics reference model and contains extensions of [DVB-MHP1.1.2] Section: 13 Graphics reference model.

OCAP 1.1 requires support for the presentation of DTVCC content. Applications can control the presentation of DTVCC through JMF using the org.ocap.media package.

The coordinate space of closed captioning is defined in [EIA-608-B] and [EIA-708-B].

The closed captioning representation SHALL be drawn over any graphics of Java application. The closed captioning layer is logically in the HgraphicsDevice. Actual implementation is manufacture dependent.

#### *15.2.2.2 OCAP HScene Management*

OCAP 1.1 extends [DVB-GEM 1.1] and adds HScene management that can be controlled by a privileged application; see Annex E.

# 16 SYSTEM INTEGRATION ASPECTS

This section contains requirements that are extensions to [DVB-GEM 1.1] Section: 14 System integration aspects and contains extensions of [DVB-MHP1.1.2] Section: 14 System integration aspects.

This section addresses issues of system integration for the OpenCable Application Platform.

This section identifies URI/URL schemes defined and supported by this profile.

## 16.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section. Section 16 (this section) of the OCAP 1.1 Profile corresponds to Section 14 of [DVB-GEM 1.1] as follows:

***Table 16–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| 16 System Integration Aspects | 14 System integration aspects | Extension |
| 16.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension |
| 16.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension |
| 16.2.1 Deviations from DVB-MHP | No Corresponding Section | OCAP-Specific Extension |
| 16.2.1.1 OCAP 1.1 Locators | 14.1 Namespace mapping | Extension |
| 16.2.1.2 Reserved Names | 14.2 Reserved names | Extension |
| 16.2.1.3 XML notation | 14.3 XML notation | Extension |
| No Corresponding Section | 14.4 Network signaling (error behavior) | Compliance |
| No Corresponding Section | 14.5 Text encoding of application identifiers | Compliance |
| No Corresponding Section | 14.6 Filename requirements | Compliance |
| 16.2.1.4 Files and File Names | 14.7 Files and file names | Extension |
| 16.2.1.5 Locators and content referencing | 14.8 Locators and content referencing | Extension |
| | 14.9 Service identification | Compliance |
| 16.2.1.6 CA System | 14.10 CA system | Extension |
| 16.2.2 Extensions to DVB (normative) | No Corresponding Section | OCAP-Specific Extension |

## 16.2 OCAP 1.1 Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

### 16.2.1 Deviations from DVB-MHP

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

#### *16.2.1.1 OCAP 1.1 Locators*

This section extends Section 14.1 of [DVB-GEM 1.1], in that GEM does not mandate any particular format of locator.

The OpenCable Application Platform allows an application to retrieve resources from a general source as defined by a domain look-up of the URI/URL via the Interactive Channel. The OCAP 1.1 Profile also provides support for referencing resources transmitted down the broadcast stream(s).

OCAP 1.1 provides the OcapLocator instead of the DvbLocator defined in [DAVIC] and [DVB-MHP1.1.2] Section 14.1. Applications SHALL use the OcapLocator.

#### *16.2.1.1.1 DAVIC Locator Compliance Statement*

The OcapLocator extends the DAVIC Locator defined in [DAVIC] and is compliant with it and its URL format with a following modification.

OCAP 1.1 uses the OCAP URL syntax for addressing the service and files within object carousels.

#### *16.2.1.1.2 OCAP 1.1 URL Naming Convention*

Like [DVB-MHP1.1.2], OCAP 1.1 extends the general Uniform Resource Locator (URL) format defined by [DAVIC] for accessing broadcast services and files within object carousels. The general format is:

```
<protocol>://<server>/<dir1>/.../<file>
```

The protocol part of the URL identifies that it is a broadcast service. The <**server**> part of the URL points to the service as the services are the basic element that is carried in the broadcast networks. The rest of the URL specifies the individual component inside a service.

The format of the OCAP URL shown in an informal notation is as follows.

```
ocap://<source_id>[.<stream_type>[,<ISO_639_language_code>]{&<stream_type>
[,<ISO_639_language_code>]}] [;<event_id>]{/<path_segments>}
ocap://<source_id>[.<stream_type>[,<index>]{&<stream_type>[,<index>]}][;<event_id>]
{/<path_segments>}
ocap://<source_id>[.+<PID>{&<PID>}][;<event_id>]{/<path_segments>}
ocap://<source_id>[.@<component_tag>{&<component_tag>}][;<event_id>]{/<path_segments>}
ocap://<source_id>[.$<component_name>{&<component_name>}][;<event_id>]{/<path_segments>}
ocap://n=<service_name>[.<stream_type>[,<ISO_639_language_code>]{&<stream_type>
[,<ISO_639_language_code>]}] [;<event_id>]{/<path_segments>}
ocap://n=<service_name>[.<stream_type>[,<index>]{&<stream_type>[,<index>]}][;<event_id>]
{/<path_segments>}
ocap://n=<service_name>[.+<PID>{&<PID>}][;<event_id>]{/<path_segments>}
ocap://n=<service_name>[.@<component_tag>{&<component_tag>}][;<event_id>]
{/<path_segments>}
ocap://n=<service_name>[.$<component_name>{&<component_name>}][;<event_id>]
{/<path_segments>}
ocap://f=<frequency>.<program_number>[.m=<modulation_format>][.<stream_type>
[,<ISO_639_language_code>]{&<stream_type>[,<ISO_639_language_code>]}][;<event_id>]
{/<path_segments>}
ocap://f=<frequency>.<program_number>[.m=<modulation_format>][.<stream_type>[,<index>]
{&<stream_type>[,<index>]}][;<event_id>]{/<path_segments>}
ocap://f=<frequency>.<program_number>[.m=<modulation_format>][.+<PID>{&<PID>}]
[;<event_id>]>]{/<path_segments>}
ocap://f=<frequency>.<program_number>[.m=<modulation_format>][.@<component_tag>
{&<component_tag>}][;<event_id>]{/<path_segments>}
ocap://f=<frequency>.<program_number>[.m=<modulation_format>][.$<component_name>
{&<component_name>}][;<event_id>]{/<path_segments>}
ocap://oobfdc.<program_number>[.<stream_type>[,<ISO_639_language_code>]
{&<stream_type>[,<ISO_639_language_code>]}][;<event_id>]{/<path_segments>}
ocap://oobfdc.<program_number>[.<stream_type>[,<index>]{&<stream_type>
[,<index>]}][;<event_id>]{/<path_segments>}
ocap://oobfdc.<program_number>[.+<PID>{&<PID>}][;<event_id>]{/<path_segments>}
ocap://oobfdc.<program_number>[.@<component_tag>{&<component_tag>}][;<event_id>]
{/<path_segments>}
ocap://oobfdc.<program_number>[.$<component_name>{&<component_name>}]
[;<event_id>]{/<path_segments>}
ocap://f=<frequency>[.m=<modulation_format>]
ocap:/<path_segments>
```

A formal specification is expressed in BNF as used in [RFC 2396]:

```
ocap_url   = ocap_scheme ":" ocap_hier_part
ocap_scheme= "ocap"
ocap_hier_part = ocap_net_path | ocap_abs_path
    (See restriction 1 below)
ocap_net_path  = "//" ocap_entity [ ocap_abs_path ]
    (See restriction 2 below)
ocap_entity= ocap_service | ocap_service_component | ocap_transport
ocap_service   = source_id | named_service | ocap_program
ocap_service_component = ocap_service [ "." program_elements ] [ ";" event_id ]
program_elements   = language_elements | index_elements | PID_elements |
      component_elements | component_tag_elements
language_elements  = stream_type [ "," ISO_639_language_code ] * ( "&" stream_type
      [ "," ISO_639_language_code ] )
    (See restriction 3 below)
index_elements = stream_type [ "," index ] * ( "&" stream_type [ "," index ] )
PID_elements   = "+" PID * ( "&" PID )
component_elements = "$" component_name * ( "&" component_name )
component_tag_elements = "@" component_tag * ( "&" component_tag )
ocap_program   = "f=" frequency "." program_number ["m=" modulation_format] |
    "oobfdc." program_number
ocap_transport     = "f=" frequency [".m=" modulation_format]
source_id  = hex_string
named_service  = "n=" service_name
service_name   = 1* (unreserved_not_dot | escaped)
         (See restriction 4 below)
component_name = 1* (unreserved | escaped)
     (See restriction 5 below)
component_tag  = hex_string
frequency  = hex_string
program_number = hex_string
modulation_format  = hex_string
stream_type= hex_string
    (See restriction 6 below)
ISO_639_language_code  = alpha alpha alpha
    (See restriction 7 below)
index  = hex_string
PID = hex_string
event_id   = hex_string
hex_string = "0x" 1*hex
hex = digit | "A" | "B" | "C" | "D" | "E" | "F" | "a" | "b" | "c" |
      "d" | "e" | "f"
digit  = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
ocap_abs_path  = "/" path_segments
      (path_segments is defined in [RFC 2396].)
    (See restriction 8 below)
path_segments  = segment *( "/" segment )
segment = *pchar *( ";" param )
param  = *pchar
pchar  = unreserved | escaped | ":" | "@" | "&" | "=" | "+" | "$" | ","
unreserved = alphanum | mark
unreserved_not_dot = alphanum | mark_not_dot
alphanum   = alpha | digit
alpha  = lowalpha | upalpha
lowalpha   = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k"
      | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" |
      "v" | "w" | "x" | "y" | "z"
upalpha = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K"
      | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" |
      "V" | "W" | "X" | "Y" | "Z"
escaped = "%" hex hex
mark   = "-" | "_" | "." | "!" | "~" | "*" | "'" | "(" | ")"
mark_not_dot   = "-" | "_" | "!" | "~" | "*" | "!" | "(" | ")"
```

This syntax is fully compliant with the generic syntax of URIs as specified in [RFC 2396] and uses the registry-based naming authority version of that recommendation. Furthermore, all generic definitions are specified in [RFC 2396] for the service_name and the component_name; See Table 16–3.

16.2.1.1.2.1 Additional Restrictions

a. When the ocap_hier_part is an ocap_abs_path, the URL refers to a file in a default object carousel within the current service.

b. If the ocap_entity is an ocap_service (i.e., not a ocap_service_component) then there shall only be one Object Carousel in the ocap service.

c. If the stream_type is an audio stream_type, then the ISO_639_language_code may be used to select a specific language track (namely audio). If the ISO_639_language_code is not present, then the default language track is selected. The default language track is defined as follows;

- if exactly one audio stream for the default language (as defined by the "User Language " preference in org.dvb.user.GeneralPreference) is signaled then that stream is the default
- if no audio streams are signaled with the default language then the first audio stream listed in the PMT is the default
- if more than one audio stream is signaled with the default language then the first such stream listed in the PMT is the default

d. The service_name may contain the characters other than "unreserved_not_dot" as defined above by encoding each such character using its ASCII representation. If the name needs to include other characters these SHALL be represented using the escaped sequence defined in [RFC 2396]. For example, the character sequence "B&B" can be expressed as "B%26B". The name in the URL SHALL be translated to UTF-8 before URL byte escaping is applied. If the character is coded in 2 bytes, 3 bytes or 4 bytes in the UTF8 character code, it shall be escaped by dividing into 8 bit escaping sequence. For example, UTF8 code "0xC080" is represented by "%C0%80" and "0xE08080" is represented by "%E0%80%80" in the OCAP URL.

e. The component_name may contain the characters other than "unreserved" as defined above by encoding each such character using its ASCII representation. If the component_name needs to include other characters these SHALL be represented using the escaped sequence defined in [RFC 2396]. For example, the character sequence "B&B" can be expressed as "B%26B". The name in the URL shall be translated to UTF-8 before URL byte escaping is applied. If the character is coded in 2 bytes, 3 bytes or 4 bytes in the UTF8 character code, it shall be escaped by dividing into 8 bit escaping sequence. For example, UTF8 code "0xC080" is represented by "%C0%80" and "0xE08080" is represented by "%E0%80%80" in the OCAP URL.

f. stream_type shall conform to those defined in Section 7.4 of the [SCTE 57] specification.

g. The encoding format of ISO_639_Language_code is UTF-8.

h. The following restrictions apply to the ocap_abs_path part of a name:

- The total length of path names, separators and filename shall be less than or equal to 254 bytes long.
- The following characters are not allowed in file names and path names: character null (0xC080), byte zero.
- The encoding of the file name is in UTF-8 (as defined in [DVB-GEM 1.1] Section 7.1.5).
- An absolute filename starts with a slash character (as indicated in the BNF above).

#### 16.2.1.1.2.2 Referencing Specific Entities

##### 16.2.1.1.2.2.1 Program Streams

Where ocap_entity is an ocap_service, the ocap service that consists of entire program streams identified by the entity is referenced.

##### 16.2.1.1.2.2.2 Program Elements

Where ocap_entity is an ocap_service_component, a single program element is referenced.

#### 16.2.1.1.2.2.3 Transports

Where ocap_entity is an ocap_transport an entire multiplex is referenced. This can be used with the org.davic.tuning.NetworkInterfaceController.tune(Locator) method.

#### 16.2.1.1.2.2.4 Files and Directories

When a path is present in a URL where the ocap_entity part identifies an ocap service, the path references an object in an object carousel within the service.

When a path is present in a URL where the ocap_entity part identifies one component of an ocap 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 ocap_entity part identifies something else than the two cases described above are not specified in this profile.

### 16.2.1.1.2.3 Resolution of Locator Elements

In cable receivers, when the CableCARD Module is present, locators shall be resolved using the SI present in the OOB signaling. In cable receivers when the CableCARD Module is absent, the in-band SI shall be used for resolution.

#### 16.2.1.1.2.3.1 Universally resolvable

Constructs listed in the Table below rely for resolution on signaling that is mandatory in all cable profiles.

***Table 16–2 - OCAP URI Universally Resolvable Constructs***

| Name | Cable | Comment |
|---|---|---|
| source_id | The source_id parameter matches either the source_id of a program stream or the service_id of an abstract service. The source_id is distinguished from the service_id by the value range. The source_id field in the VCM_structure of the Short Form Virtual Channel Table (Profile 1 through 5) or the source_id field in Long Form Virtual Channel Table (Profile 5 and 6) as defined in [SCTE 65]. If both are present, the Short Form version shall be used. If no CableCARD Module is present, then the source_id field in the Cable Virtual Channel Table as used in the in-band SI shall be used. The service_id is a 24-bit value defined in Section 11.2.2.3.14. If the service_id is specified, the abstract service identified by the service_id is indicated. | |
| stream_type | The first program element matching that stream_type. The stream_types are defined in the stream_type assignments Table of [ISO 13818-1] and in the Stream_Type Codes Table of [SCTE 54]. | Note that the specified stream is not guaranteed to be decoded if [HOST 2.0] does not support decoding it. |
| ISO 639 language code | The first audio program element where there is a match between the specified ISO 639-2 3-character language code and the contents of the ISO 639 descriptor. | |

#### 16.2.1.1.2.3.2 Environment specific

Constructs listed in the Table below are those where the underlying signaling is not required to be present in all cable profiles.

***Table 16–3 - OCAP URI Environment Specific Constructs***

| Name | Cable | Comment |
|---|---|---|
| service_name | If the service information contains a Long-form Virtual Channel Table, the short_name from that Table is translated to a UTF-8 string and compared with the UTF-8 representation of service_name. Otherwise, if the service information contains a Source Name Sub-table in the Network Text Table, each source_name component with mode less than 0x40 is translated to a UTF-8 string according to its mode and byte string and compared with the UTF-8 representation of service_name. Components of source_name using format-effector modes are ignored in the comparison. Otherwise the service_name is not resolvable. | Use of this in cable assumes the MSO ensures these names are uniquely correlated with source_ids and DSG application_ids in their network. These names are not interchangeable between cable networks. |
| component_tag | A component_tag value in one of the Stream Identifier Descriptors located in the inner descriptor loop of the TS_program_map_section associated with the Virtual Channel identified. | Where component tag is used with an environment specific virtual channel identification (e.g., short_name) then it is also environment specific. |
| component_name | The component name string in the Component Name Descriptor located in the inner descriptor loop of the TS_program_map_section associated with the Virtual Channel (see below). | This identifier can only be used with cable systems supporting Profiles 4, 5 and 6 of [SCTE 65]. |
| event_id | The event_id identifier shall correspond to the event_ID in the Aggregate Event Information Table (AEIT) as defined in [SCTE 65]. | Event identifiers shall be scoped by a Virtual Channel identifier. This identifier may only be resolved in systems supporting Profiles 4,5,6. |

The component_name in the PMT is represented as a Multiple String Structure with each set of string components associated with a specific language. The set of string components corresponding to language code "eng" are selected, and decompressed for comparison. Each PMT component_name string component with mode less than 0x40 is translated to a UTF-8 string according to its mode and byte string and compared with the UTF-8 representation of the component_name extracted from the locator. Components of the PMT component_name using format-effector modes are ignored in the comparison.

#### 16.2.1.1.2.3.3 1Physical constructs

Constructs listed in the Table below are specific to a particular environment or cable head-end.

**Note**: Applications should not include hard-coded values of these. Locators using them are intended to be dynamically constructed in the ocap receiver based on locally accurate information; (e.g., as would be returned by org.ocap.si.PMTElementaryStreamInfo.getElementaryPID()).

***Table 16–4 - OCAP URI Physical Layer Constructs***

| Name | Cable | Comment |
|---|---|---|
| frequency | The frequency is a 32-bit hex value in hertz, which can be used to tune to a service that is only defined within an inband PAT and PMT. | |
| oobfdc | Locates the OOB forward data channel. | Can be used to locate an OOB object carousel. |
| program_number | A 16-bit value as specified in [ISO 13818-1]. | |
| modulation_format | Equal to the modulation_format field of the MMS record as specified by [SCTE 65] Table 5.9. When this optional field is not included in a locator and an OOB SI entry for the frequency and subsequently the modulation format is not signaled, a default value of QAM256 is given. | |
| PID | In this case the program element is indicated by the PID. | |
| Index in PMT | In this case the program element is the indexed program element matching that stream_type. The index specifies the ordinal number of the elementary streams that have same stream_type in the PMT. The first elementary stream of them is index = 0. | If multiple MPEG PES of the same stream_type are present in the program, then the index can be used to select the first, second, third, and so forth. |

#### *16.2.1.1.3 Examples of OCAP 1.1 URL Usage*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 14.1 Namespace mapping.

The following Table shows example usage of OCAP 1.1 locators.

***Table 16–5 - OCAP 1.1 URL Examples***

| Example | Description |
|---|---|
| ocap://0x1234 | The service for source_id 0x1234 |
| ocap://0x1234.0x2 | The first [ISO 13818-2] video stream for source_id 0x1234 |
| ocap://0x1234.0x81 | The first [ATSC A/53E] audio stream for source_id 0x1234 |
| ocap://n=MOVIE | The service with source_name or short_name MOVIE |
| ocap://n=MOVIE.0x81,spa | Spanish language [ATSC A/53E] audio track for the service with source_name or short_name MOVIE |
| ocap://0x1234/<path>/<filename> | The file of <filename> in the DSM-CC object carousel in the service for source_id 0x1234. |
| ocap://0x1234.+0x56&0x78;0xBC | Two program elements that has PID=0x56 (for video) and 0x78 (for audio) in the service for source_id 0x1234. It is also restricted by event_id 0xBC. |
| ocap://f=FREQ.m=QAM256 | The service for frequency FREQ and modulation format QAM256 |

#### 16.2.1.2 Reserved Names

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 14.2 Reserved names.

File names starting with the characters 'ocap.' are reserved for use as files defined in this profile.

Authors SHALL NOT use file names with this form to avoid collisions with OCAP 1.1 defined files.

#### 16.2.1.3 XML notation

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 14.3 XML notation.

The PublicLiteral is used for identifying the type of the XML file. For document types specified in this profile, the PublicLiteral SHALL have the following syntax:

```
"-//OCAP//DTD " <document type> " " <version_number> "//EN"
```

where <document type> and <version_number> are as specified in Section 14.3 of [DVB-MHP1.1.2].

#### 16.2.1.4 Files and File Names

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 14.7 Files and file names.

The OCAP 1.1 Profile defines how applications can use the names of files in order to access content held in files. Like the [DVB-MHP1.1.2], it is intentionally silent about the file systems and file system name-spaces of OCAP 1.1 terminals except as defined below:

- When an OCAP 1.1 application starts, the file system where that application is carried is mounted into the file system namespace for the OCAP 1.1 terminal concerned. For an OCAP application, Section 11.5.1 [DVB-MHP1.1.2] defines that creating a new instance of `java.io.File(".")` results in a reference to the base directory of the application. This base directory MAY be a sub-directory within this file system. The method java.io.lastModified() SHALL return an integer in accordance with [DVB-MHP1.1.2], Section 11.5.1.1, as modified by [DVB-GEM 1.1], Section 11.5.1.
- When an application is run from storage, a) the OCAP host device shall cause the set of successfully stored files from that application to appear in its file-system in the same hierarchy as defined in the ADF. b) the relative path "." shall refer to the directory in the file-system of the OCAP host device where any files stored from the base directory of the OCAP-J application would appear. c) OCAP host devices shall not automatically make any connection to the carousel from which the application was originally stored.

  NOTE 1: Hence if the base directory of an OCAP-J application contains the file "foo.txt" and this is listed in the ADF then when that application is stored and runs from storage, if the application calls new java.io.File("./foo.txt") then this will successfully access the copy of "foo.txt" that was stored and will neither fail nor cause an access to the file system from which the application was originally stored.

  NOTE 2: If an application running from storage wishes to access the carousel from which it was originally stored, the application is responsible for using the org.dvb.dsmcc API (and perhaps the tuning API) to cause that carousel to appear in the file-system of the OCAP host device. The application may then obtain a reference to the place in the OCAP host device file-system where the top level directory of the carousel appears by calling ServiceDomain.getMountPoint(). Applications running from storage cannot use relative paths to access files in explicitly mounted object carousels.

  NOTE 3: If an unbound application is down loaded and stored from an in-band carousel, access to that in-band carousel when the application is running from storage may require interrupting any video and audio currently being watched by the end-user of the OCAP host device.

- When an application is running from storage and the object carousel from which it was originally stored has been attached (using the ServiceDomain.attach() method), this carousel shall appear at a different location in the file system namespace of the OCAP host device from the stored files (i.e., the copy of a file in a carousel and the copy in storage shall always have different paths in the file-system hierarchy of the OCAP host device).
- Any given absolute path in the file-system hierarchy of the OCAP host device shall always refer to the same actual copy of a file regardless of which API that path is used with. In the context of any single application, any given relative path shall always refer to the same actual copy of a file regardless of which API that path is used with.
- OCAP 1.1 applications that have requested the right to access persistent storage, and had this right granted, MAY access the persistent file namespace. For OCAP applications, the top level directory of this namespace MAY be found from the system property, `dvb.persistent.root`.
- OCAP 1.1 applications MAY have the ability to mount additional file systems into the file system namespace of the OCAP 1.1 terminal concerned. OCAP applications MAY use the attach() method on the org.dvb.dsmcc.ServiceDomain class in order to attach an object carousel as an additional file system. Any locator that is an OCAP 1.1 Service is a valid input to the attach() method. In all other methods, using an OCAP locator including the ocap_abs_path part of the name part of the syntax SHALL NOT mount the specified object carousel file system.
- Conformant OCAP 1.1 applications SHALL NOT attempt to access files or file systems outside what is allowed by this profile. The consequences should they attempt to do this are undefined and implementation dependent. Platforms MAY choose to limit the access rights of OCAP applications through use of platform security mechanisms, e.g., `java.io.FilePermission`.
- References to content carried in files SHALL either be done using names of files encoded in text or using 'file:' URLs as defined in [RFC 1738]. OCAP application SHALL transform file name encoded in 'ocap:' URLs to 'file:' URLs before use. For OCAP applications, file names SHALL be encoded in Java String objects and 'file:' URLs SHALL be encoded in instances of java.net.URL.

#### *16.2.1.5 Locators and content referencing*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 14.8 Locators and content referencing.

Table 16–6 lists the types of entity that MAY be addressed by locators in the OpenCable Application Platform. The Table also calls out the text representation, the URI/URL naming convention, for each entity.

***Table 16–6 - OCAP 1.1 Addressable Entities***

| Entity | Text Representation |
|---|---|
| Transport stream | OCAP locator including *"<ocap_transport>"* element |
| Network | No standardized text representation |
| OCAP 1.1 Service | OCAP locator including "<ocap_service>" element |
| Generic Service | No standardized text representation unless also a OCAP 1.1 Service |
| OCAP 1.1 Event | OCAP locator including "ocap_service_component" element and "event_id" element. |
| MPEG Elementary Stream | ocap://<ocap_service>.<program_element> |
| File | file: URL as defined in [RFC 1738]1<br>OCAP locator including "ocap_abs_path" element2 |
| Directory | file: URL as defined in [RFC 1738]1<br>OCAP locator including "ocap_abs_path" element2 |

| Entity | Text Representation |
|---|---|
| Drip feed decoder | "dripfeed://" |

Table Note 1. The host name part of a "file:" URL SHALL always be the empty string. These URLs are always in the namespace of the receiver.

Table Note 2. OCAP locators including the "ocap_abs_path" element may be returned by OCAP 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.

This profile does not require support for addressing any other type of entity, either by locator or URI/URL, beyond those specified in the above Table.

#### *16.2.1.5.1 Transport stream*

This entity is represented by a locator that includes "ocap_transport" element.

#### *16.2.1.5.2 Network*

OCAP 1.1 does not provide a textual representation for accessing this entity.

#### *16.2.1.5.3 OCAP 1.1 Service*

This entity is a program defined as a source in the [SCTE 65]. In practice, a source consists of zero or one video elemental streams, one or more audio elemental streams, and zero or more data elemental streams.

An OCAP Service Locator can be created using an ocap_service term, see Section 16.2.1.1.2. An OCAP Service Locator can be used to create a javax.tv.service.navigation.LocatorFilter. Either an OCAP service Locator or a LocatorFilter can be used to discover the corresponding Service object by calling the getService() and filterServices() methods respectively, in an SIManager object that was assigned to a call to the javax.tv.service.SIManager.createInstance() method. When either the afore mentioned getService() or filterServices() calls are made, if the Service is not in the SI database and the Locator is valid, the OCAP implementation SHALL create the Service and return it. The implementation is not required to place a Service into the SI database if it was not discovered by signaled SI. If an OCAP Service object is tuned to using the select(Service) method of an object of type javax.tv.service.select.ServiceContext, but the service cannot be found because the Locator consists of a frequency/program_number pair but the program_number is not signaled in the in-band PAT, then the implementation SHALL generate a SelectionFailedEvent.

When the SIManager.getService(Locator) method is called with a Locator parameter that references a hidden channel, a Service object for that channel SHALL be returned; see Annex T.2.2.7.7 for SI mapping. The javax.tv.service.navigation.LocatorFilter.accept(Service) method SHALL always return false when passed a Service instance that represents a hidden channel.

16.2.1.5.3.1 OCAP 1.1 Implementation Constructors for OCAP 1.1 Service Locators

When an application calls javax.tv.service.Service.getLocator(), the OCAP 1.1 implementation SHALL construct the OCAPLocator as follows:

a. For services that have been created from a frequency / program_number pair, or "oobfdc" / program_number pair, as described in Section 16.2.1.5.3, one of the OcapLocator constructors taking a frequency parameter is used. This rule applies even in the case that the source_id can be identified from the virtual channel Table.

b. For services that have been created by any other means, one of the OcapLocator constructors taking a

sourceID parameter is used. For a broadcast service, the sourceID is the 16-bit source_id provided in the signaling for the service. For an abstract service, the sourceID is the 24-bit service_id that identifies the service.

#### *16.2.1.5.4 Generic Service*

This entity is represented by a locator that facilitates generic Service Information provided by JavaTV and Java Media Framework APIs. This is a non-streaming service.

Any locator that is an OCAP 1.1 Service is also valid as a Generic Service locator.

#### *16.2.1.5.5 OCAP 1.1 Event*

This entity, represented by a locator, includes the "event_id" elements. OCAP 1.1 events will be resolvable only if an AEIT of [SCTE 65] is available.

#### *16.2.1.5.6 MPEG Elementary Stream*

This entity is represented by a locator associated with MPEG elementary streams.

#### *16.2.1.5.7 File*

This entity is represented by a locator that references files. It is used to access data on a currently mounted file system and can vary between OCAP 1.1 receivers.

`The path_segments` in the "ocap://<ocap_service>/<path_segments>" expression is defined in [RFC 2396].

Refer to Section 16.2.1.4 regarding files and file names.

#### *16.2.1.5.8 Directory*

This entity is represented by a locator that indicates a directory. It is used to access a directory on a currently mounted file system and can vary between OCAP 1.1 receivers.

#### *16.2.1.5.9 Drip feed decoder*

The `dripfeed://` locator format is used to request an instance of `org.dvb.media.DripFeedDataSource`. More information concerning this locator type can be found in Annex N, Streaming Media API Extensions of the [DVB-MHP1.1.2].

### *16.2.1.6 CA System*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 14.10 CA system.

This section complies with and extends Section 14.10 of [DVB-GEM 1.1] in that both GEM and OCAP 1.1 do not support the DVB CAM and CI. All CA transactions occur implicitly and are not exposed by the OCAP APIs except in the instance of an exception or event. The MHP Host clear/clear_replace does not apply in OCAP because the CableCARD device can cause the Host to tune using CableCARD interface/Host resources as defined by [CCIF 2.0].

A CableCARD has the ability to descramble streams in at least one service at a time. The service to be descrambled is specified by a ca_pmt APDU from the Host to the CableCARD; see [CCIF 2.0] for ca_pmt APDU definition. However, the OCAP implementation may request descrambling of streams in several services in a transport stream to the CableCARD, and it may also request descrambling of streams in several transport streams if the host has multiple tuners. In such cases, it SHALL be guaranteed that streams in current services successfully selected by ServiceContext.select() calls are descrambled. If the requested number of services exceeds the ability of the

CableCARD, the ServiceContext.select() call SHALL fail. Descrambling of streams outside the selected services (played by a JMF player or attaching of a DSMCC service domain) SHALL be allowed only if it doesn't prevent descrambling of any ServiceContext selected service. If calls to play streams outside the selected service occur after ServiceContext service selection, the later calls SHALL be prioritized to be descrambled. When a new call of the ServiceContext.select() method occurs, descrambling of some streams outside any ServiceContext selected service may be terminated to start new descrambling.

In case of using a single-stream CableCARD, it SHALL be guaranteed that streams in the scrambled service selected by the last successful call to the ServiceContext.select() method are always descrambled. Where there is more than one service context presenting a broadcast service, in the event of a conflict for resources in the CableCARD, the most recent successful service selections SHALL be given priority. Streams outside the selected service that are played by JMF player or attaching of a DSMCC service domain, SHALL be descrambled only if descrambling of the streams in the selected service is unnecessary. In such a case, the streams played by the last call SHALL be descrambled.

The implementation sends a ca_pmt APDU to the CableCARD for several reasons, such as a tune to a new service, and the CableCARD responds with a ca_pmt_reply. The CableCARD may also send an unsolicited ca_pmt_reply APDU to the implementation when CA status changes. The ca_pmt_reply APDU contains a program_number field indicating the service, and a ca_enable field for each elementary stream. The ca_enable field indicates a descrambling status for the elementary stream. Possible values include:

- 0x01 Descrambling possible with no extra conditions.
- 0x02 Descrambling possible under conditions. An entitlement session external to CableCARD and Host APDUs MAY change this status in a subsequent unsolicited ca_pmt_reply APDU.
- 0x03 Descrambling possible under technical conditions. A resource acquisition session external to CableCARD and Host APDUs MAY change this status in a subsequent unsolicited ca_pmt_reply APDU.
- 0x71 Descrambling not possible for entitlement reasons, e.g., the selected program is not entitled and is not available for purchase.
- 0x72 Reserved.
- 0x73 Descrambling not possible for technical reasons, e.g., CableCARD resources needed cannot be acquired via an external session.

The value returned in the ca_enable field is mapped to various JavaTV and JMF events based on the API used by an application to select elementary streams. When the ServiceContext.select(Service) method is called and while the ServiceContext is presenting based on that call, if any elementary stream cannot be descrambled an alternative content event is generated. In this case the implementation SHALL execute the flowchart in Figure 16–1 whenever it receives a ca_pmt_reply APDU from the CableCARD:

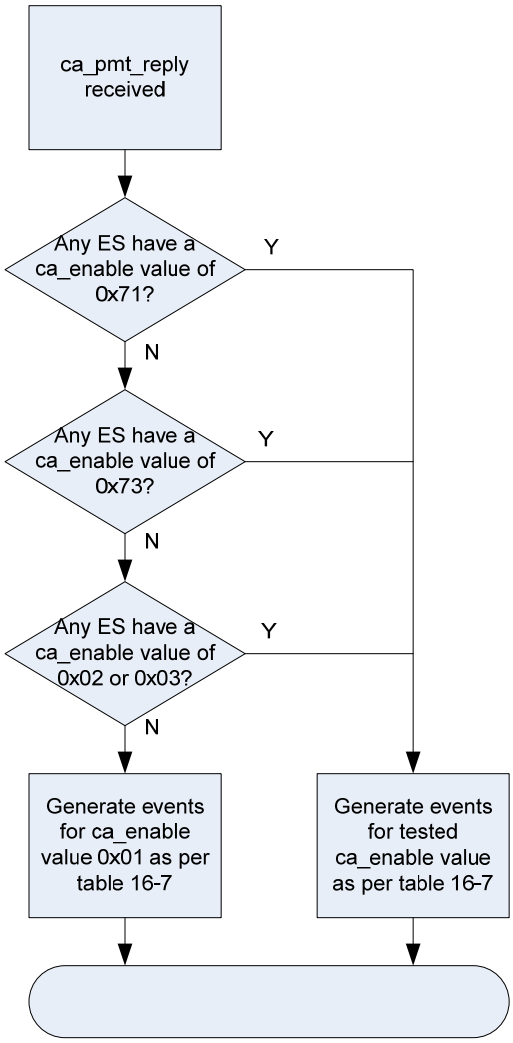



***Figure 16–1 - CA event generation for ServiceContext.select(Service)***

When the ServiceContext(Locator []) method is used, presentation can occur even if one or more requested service components cannot be presented; see [9] DVB-MHP 1.0.3 annex section A.5.1.2. When the ServiceContext.select(Locator []) method is called and while the ServiceContext is presenting based on that call, the implementation SHALL execute the flowchart in Figure 16–2 whenever it receives a ca_pmt_reply APDU from the CableCARD.

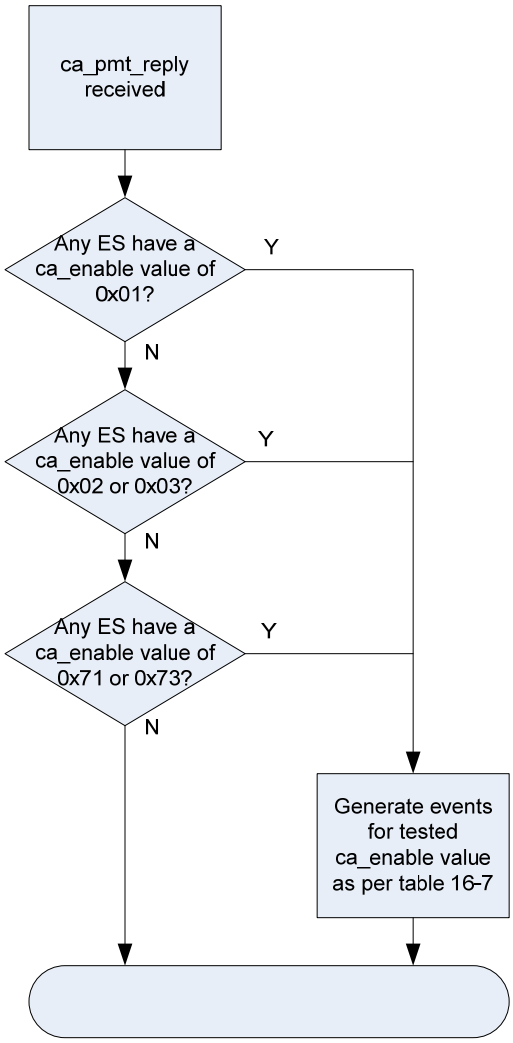



***Figure 16–2 - CA event generation for ServiceContext.select(Service)***

When a discrete Player without ServiceContext association is started and while it is presenting, the implementation SHALL follow the same logic as for the ServiceContext.select(Service) method; seeFigure 16–1. However, only events in the Player column of Table 16–7 are generated.

When a ServiceContext.select method is called the implementation MAY create new ServiceMediaHandler instances for the ServiceContext; see [9] DVB-MHP 1.0.3 section 11.6.2. Because of this possibility, listeners added to ServiceContext associated ServiceMediaHandler instances before a select method call MAY not be listening to the correct set of ServiceMediaHandler instances after a select method call.

When a ServiceContext generates an AlternativeContentEvent it will be in the presenting state as per [Java TV]; see AlternativeContentEvent javadoc. Any ServiceMediaHandlers returned by the getServiceContentHandlers method of a ServiceContext that is presenting alternative content SHALL NOT be in the STARTED state.

***Table 16–7 - ca_enable field values returned in a ca_pmt_reply***

| ca_enable field value returned in a ca_pmt_reply | ServiceContext Event generated | ServiceMediaHandler or discrete Player Event generated |
|---|---|---|
| 0x01 Descrambling possible. | NormalContentEvent | NormalMediaPresentationEvent |

| ca_enable field value returned in a ca_pmt_reply | ServiceContext Event generated | ServiceMediaHandler or discrete Player Event generated |
|---|---|---|
| 0x02 Descrambling possible under purchase conditions. | AlternativeContentEvent | AlternativeMediaPresentationEvent |
| 0x03 Descrambling possible under technical conditions. | AlternativeContentEvent | AlternativeMediaPresentationEvent |
| 0x71 Descrambling not possible, no purchase possible. | AlternativeContentEvent | AlternativeMediaPresentationEvent |
| 0x73 Descrambling not possible, no resources available. | AlternativeContentEvent | AlternativeMediaPresentationEvent |

**Note:** In case unbound applications and/or the Monitor Application are launched from scrambled DSMCC object carousels, a new descrambling of another service may cause detaching of such DSMCC object carousels.

### 16.2.2 Extensions to DVB (normative)

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

#### *16.2.2.1 Mandatory Ordinary Keycodes*

The Mandatory Ordinary Keycodes are specific key codes that can't be filtered by OCAP event filtering. OCAP defines event filtering in Annex K.2.1 to modify a key code and a destination that are contained in an event. However, it is guaranteed that an event that contains one of Mandatory Ordinary Keycodes is delivered to an original destination application without modification of the original event code (i.e., the original Mandatory Ordinary Keycodes). Mandatory Ordinary Keycodes are a part of the minimum set of key codes. The set of Mandatory Ordinary Keycodes is defined by Table 25–5.

Mandatory Ordinary Keycodes have the following characteristics:

- An application can reserve an AWTEvent and a UserEvent that contains one of Mandatory Ordinary Keycodes, either exclusively, or non-exclusively (see Annex K).
- The Monitor Application can't filter either an AWTEvent or a UserEvent that contains one of Mandatory Ordinary Keycodes. See the EventManager.setFilteredRepository() method in Annex K.

# 17 DETAILED PLATFORM PROFILE DEFINITION

This section contains requirements that are extensions to [DVB-GEM 1.1] Section: 15 Detailed platform profile definitions and are extensions of [DVB-MHP1.1.2] Section: 15 Detailed platform profile definitions.

The DVB-GEM defines 3 distinct profiles, the characteristics of which are detailed in Section 16 of DVB-GEM (Section 15 of[DVB-MHP1.1.2]). OCAP 1.1 specifies a single profile. It incorporates the profiles defined by DVB-GEM as specified in Section 17.2.1.

## 17.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that does not correspond to any [DVB-GEM 1.1] Section.

Section 17 (this section) of the OCAP 1.1 Profile corresponds to Section 15 of [DVB-MHP1.1.2] as follows:

***Table 17–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 17 Detailed Platform Profile Definition | 15 Detailed platform profile definitions | Extension | 15 Detailed platform profile definitions | Extension |
| 17.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 17.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| No Corresponding Section | 15.0 General | Compliance | | |
| No Corresponding Section | 15.1 PNG - restrictions | Compliance | 15.1 PNG - restrictions | Compliance |
| 17.2.1 Deviations from DVB-MHP | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 17.2.1.1 Minimum Platform Profile | 15.2 Minimum media formats supported by DVB-J APIs | Extension | 15.2 Minimum media formats supported by DVB-J APIs | Extension |
| No Corresponding Section | 15.3 JPEG - restrictions | Compliance | 15.3 JPEG - restrictions | Compliance |
| 17.2.1.2 Locale Support | 15.4 Locale support | Extension | 15.4 Locale support | Extension |
| No Corresponding Section | 15.5 Video raster format dependencies | Compliance | 15.5 Video raster format dependencies | Compliance |
| No Corresponding Section | 15.6 Functional equivalents | Compliance | | |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| No Corresponding Section | 17 Internet Access Client<br>Note: Section 15 table 7 indicates that GEM Clause 17 Internet Access Client is optional in all profiles. | Not Supported | 17 Internet Access Client | Not supported |

## 17.2 OCAP 1.1 Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

### 17.2.1 Deviations from DVB-MHP

This subsection contains OCAP-specific requirements that does not correspond to any [DVB-GEM 1.1] Section.

Section 17 (this section) of the OCAP 1.1 Profile deviates from Section 15.2 of [DVB-MHP1.1.2] as follows:

#### *17.2.1.1 Minimum Platform Profile*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 15.2 Minimum media formats supported by DVB-J APIs and are extensions of [DVB-MHP1.1.2] Section: 15.2 Minimum media formats supported by DVB-J APIs.

The OCAP 1.1 Profile deviates from Section 15.2 of [DVB-MHP1.1.2] as follows:

Support for the Interactive Profile 1 in Section 16, Table 6 of [DVB-GEM 1.1] (Section 15, Table 65 of [DVB-MHP1.1.2]) is required for OCAP 1.1, with the additional requirements specified in Section 8.2.2.2 of this document. The definition of the Interactive Profile 1 includes all features and properties of the Enhanced Broadcast Profile 1.

#### *17.2.1.2 Locale Support*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: 15.4 Locale support and are extensions of [DVB-MHP1.1.2] Section: 15.4 Locale support.

The OCAP 1.1 Profile extends Section 15.4 of [DVB-GEM 1.1] as follows:

# 18 REGISTRY OF CONSTANTS

This section contains requirements that are extensions to [DVB-GEM 1.1] Section: 16 Registry of constants and are extensions of [DVB-MHP1.1.2] Section: 16 Registry of Constants.

This section itemizes the system constants required by the OpenCable Application Platform. This section specifies the values of the public final static symbols from the various Java APIs.

## 18.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

Section 18 (this section) of the OCAP 1.1 Profile corresponds to Section 16 of [DVB-MHP1.1.2] and [DVB-GEM 1.1] as follows:

***Table 18–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 18 Registry of Constants | 16 Registry of constants | Extension | 16 Registry of Constants | Extension |
| 18.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 18.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 18.2.1 Deviations from the DVB-MHP Specification | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 18.2.1.1 System Constants | 16.1 System constants | Compliance | 16.1 System constants | Extension |
| 18.2.1.2 JAVA Constants | 16.2 DVB-J constants | Compliance | 16.2 DVB-J constants | Extension |
| 18.2.1.3 OCAP-Specific JAVA Constants | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |

## 18.2 OCAP 1.1 Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

### 18.2.1 Deviations from the DVB-MHP Specification

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

#### *18.2.1.1 System Constants*

This subsection is compliant with [DVB-GEM 1.1] Section: 16.1 System constants and are extensions of [DVB-MHP1.1.2] Section: 16.1 System constants.

In Table 69 of [DVB-MHP1.1.2]: Profile encoding is not relevant to the OCAP 1.1 Profile, because OCAP 1.1 does not subscribe to the DVB-MHP application profile model.

#### *18.2.1.2 JAVA Constants*

This subsection is compliant with [DVB-GEM 1.1] Section: 16.2 DVB-J constants and contains extensions of [DVB-MHP1.1.2] Section: 16.2 DVB-J constants.

The OCAP 1.1 Profile complies with Section 16.2 of [DVB-GEM 1.1].

#### *18.2.1.3 OCAP-Specific JAVA Constants*

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

All constants are public final. These OCAP 1.1-specific JAVA constants are set in the following packages:

In org.ocap.application.AppPattern:

```
int ALLOW                  = 1
int DENY                   = 2
int ASK                = 3
```

In org.ocap.application.OcapAppAttributes:

```
int AUTOSTART              = 1
int PRESENT                = 2
int DESTROY                = 3
int KILL               = 4
int PREFETCH               = 5
int REMOTE             = 6
int OCAP_J             = 1
```

In org.ocap.net.OCRCInterface:

```
int SUBTYPE_CATV_DOCSIS = 1
int SUBTYPE_CATV_OOB = 2
```

In org.ocap.si.DescriptorTag:

```
short VIDEO_STREAM = 0x02;
short AUDIO_STREAM = 0x03;
short HIERARCHY = 0x04;
short REGISTRATION = 0x05;
short DATA_STREAM_ALIGNMENT = 0x06;
short TARGET_BACKGROUND_GRID = 0x07;
short VIDEO_WINDOW = 0x08;
short CA = 0x09;
short ISO_639_LANGUAGE = 0x0A;
short SYSTEM_CLOCK = 0x0B;
short MULTIPLEX_UTILIZATION_BUFFER = 0x0C;
short COPYRIGHT = 0x0D;
short MAXIMUM_BITRATE = 0x0E;
short PRIVATE_DATA_INDICATOR = 0x0F;
short SMOOTHING_BUFFER = 0x10;
short STD = 0x11;
short IBP = 0x12;
short CAROUSEL_IDENTIFIER = 0X13
short ASSOCIATION_TAG = 0x14;
short APPLICATION = 0x00;
short APPLICATION_NAME = 0x01;
short TRANSPORT_PROTOCOL = 0x02;
short DVB_J_APPLICATION = 0x03;
short DVB_J_APPLICATION_LOCATION = 0x04;
short EXTERNAL_APPLICATION_AUTHORISATION = 0x05;
short IPV4_ROUTING = 0x06;
short IPV6_ROUTING = 0x07;
short APPLICATION_ICONS = 0x0B;
short PRE_FETCH = 0x0C;
short DII_LOCATION = 0x0D;
short STREAM_IDENTIFIER = 0x52;
```

```
short PRIVATE_DATA_SPECIFIER = 0x5F;
short DATA_BROADCAST_ID = 0x66;
short APPLICATION_SIGNALING = 0x6F;
short SERVICE_IDENTIFIER = 0x0D;
short LABEL = 0x70;
short CACHING_PRIORITY = 0x71;
short CONTENT_TYPE = 0x72;
short STUFFING = 0x80;
short AC3_AUDIO = 0x81;
short CAPTION_SERVICE = 0x86;
short CONTENT_ADVISORY = 0x87;
short REVISION_DETECTION = 0x93;
short TWO_PART_CHANNEL_NUMBER = 0x94;
short CHANNEL_PROPERTIES = 0x95;
short DAYLIGHT_SAVINGS_TIME = 0x96;
short EXTENDED_CHANNEL_NAME_DESCRIPTION = 0xA0;
short SERVICE_LOCATION = 0xA1;
short TIME_SHIFTED_SERVICE = 0xA2;
short COMPONENT_NAME = 0xA3;
short MAC_ADDRESS_LIST = 0xA4;
short ATSC_PRIVATE_INFORMATION = 0xAD;
```

In org.ocap.ui.event.OCRcEvent:

```
int OCRC_FIRST = 600
int VK_RF_BYPASS = 600
int VK_EXIT = 601
int VK_MENU = 602
int VK_NEXT_DAY = 603
int VK_PREV_DAY = 604
int VK_APPS = 605
int VK_LINK = 606
int VK_LAST = 607
int VK_BACK = 608
int VK_FORWARD = 609
int VK_ZOOM = 610
int VK_SETTINGS = 611
int VK_NEXT_FAVORITE_CHANNEL = 612
int VK_RESERVE_1 = 613
int VK_RESERVE_2 = 614
int VK_RESERVE_3 = 615
int VK_RESERVE_4 = 616
int VK_RESERVE_5 = 617
int VK_RESERVE_6 = 618
int VK_LOCK = 619
int VK_SKIP = 620
int VK_LIST = 621
int VK_LIVE = 622
int VK_ON_DEMAND = 623
int VK_PINP_MOVE = 624
int VK_PINP_UP = 625
int VK_PINP_DOWN = 626
int OCRC_LAST = 626
```

In org.ocap.system.event.ErrorEvent:

```
int APP_INFO_GENERAL_EVENT = 0x0C000000;
int APP_REC_GENERAL_ERROR = 0x2C000000;
int APP_REC_JAVA_THROWABLE = 0x24000001;
int APP_CAT_GENERAL_ERROR = 0x3C000000;
int SYS_INFO_GENERAL_EVENT = 0x04000000;
int SYS_REC_GENERAL_ERROR = 0x24000000;
int SYS_CAT_GENERAL_ERROR = 0x34000000;
int SYS_CAT_JAVA_THROWABLE = 0x34000001;
```

In org.ocap.system.event.RebootEvent:

```
int REBOOT_BY_IMPLEMENTATION = 0x44000000;
int REBOOT_FOR_UNRECOVERABLE_SYS_ERROR = 0x44000001;
int REBOOT_FOR_UNRECOVERABLE_HW_ERROR = 0x44000002;
int REBOOT_BY_TRUSTED_APP = 0x44000003;
```

In org.ocap.system.event.ResourceDepletionEvent:

```
int RESOURCE_SYS_MEM_DEPLETED = 0x54000000;
int RESOURCE_VM_MEM_DEPLETED = 0x54000001;
```

```
int RESOURCE_CPU_BANDWIDTH_DEPLETED = 0x54000002;
int RESOURCE_RC_BANDWIDTH_DEPLETED = 0x54000003;
```

In org.ocap.system.event.SystemEvent:

```
int BEGIN_SYS_INFO_EVENT_TYPES = 0x00000000;
int BEGIN_SYS_INFO_RESERVED_EVENT_TYPES = 0x04000000;
int END_SYS_INFO_EVENT_TYPES = 0x07FFFFFF;
int BEGIN_APP_INFO_EVENT_TYPES = 0x08000000;
int BEGIN_APP_INFO_RESERVED_EVENT_TYPES = 0x0C000000;
int END_APP_INFO_EVENT_TYPES = 0x1FFFFFFF;
int BEGIN_SYS_REC_ERROR_EVENT_TYPES = 0x20000000;
int BEGIN_SYS_REC_ERROR_RESERVED_EVENT_TYPES = 0x24000000;
int END_SYS_REC_ERROR_EVENT_TYPES = 0x27FFFFFF;
int BEGIN_APP_REC_ERROR_EVENT_TYPES = 0x28000000;
int BEGIN_APP_REC_ERROR_RESERVED_EVENT_TYPES = 0x2C000000;
int END_APP_REC_ERROR_EVENT_TYPES = 0x2FFFFFFF;
int BEGIN_SYS_CAT_ERROR_EVENT_TYPES = 0x30000000;
int BEGIN_SYS_CAT_ERROR_RESERVED_EVENT_TYPES = 0x34000000;
int END_SYS_CAT_ERROR_EVENT_TYPES = 0x37FFFFFF;
int BEGIN_APP_CAT_ERROR_EVENT_TYPES = 0x38000000;
int BEGIN_APP_CAT_ERROR_RESERVED_EVENT_TYPES = 0x3C000000;
int END_APP_CAT_ERROR_EVENT_TYPES = 0x3FFFFFFF;
int BEGIN_SYS_REBOOT_EVENT_TYPES = 0x40000000;
int BEGIN_SYS_REBOOT_RESERVED_EVENT_TYPES = 0x44000000;
int END_SYS_REBOOT_EVENT_TYPES = 0x47FFFFFF;
int BEGIN_SYS_RES_DEP_EVENT_TYPES = 0x50000000;
int BEGIN_SYS_RES_DEP_RESERVED_EVENT_TYPES = 0x54000000;
int END_SYS_RES_DEP_EVENT_TYPES = 0x57FFFFFF;
int BEGIN_SYS_DNLD_EVENT_TYPES = 0X58000000;
int END_SYS_DNLD_EVENT_TYPES = 0X58FFFFFF;
```

In org.ocap.system.event.SystemEventManager:

```
int ERROR_EVENT_LISTENER = 0x0;
int REBOOT_EVENT_LISTENER = 0x1;
int RESOURCE_DEPLETION_EVENT_LISTENER = 0x2;
```

In org.ocap.hardware.Host:

```
int FULL_POWER = 1
int LOW_POWER = 2
```

In org.ocap.storage.StorageManagerEvent

```
int STORAGE_PROXY_ADDED = 1
int STORAGE_PROXY_REMOVED = 2
int STORAGE_PROXY_CHANGED = 3
```

In org.ocap.storage.StorageProxy

```
byte READY = 0;
byte OFFLINE = 1;
byte BUSY = 2;
byte UNSUPPORTED_DEVICE = 3;
byte UNSUPPORTED_FORMAT = 4;
byte UNINITIALIZED = 5;
byte DEVICE_ERROR = 6;
byte NOT_PRESENT = 7
```

In org.ocap.media.ClosedCaptioningAttribute:

```
int CC_ATTRIBUTE_PEN_FG_COLOR = 0;
int CC_ATTRIBUTE_PEN_BG_COLOR = 1;
int CC_ATTRIBUTE_PEN_FG_OPACITY = 2;
int CC_ATTRIBUTE_PEN_BG_OPACITY = 3;
int CC_ATTRIBUTE_FONT_STYLE = 4;
int CC_ATTRIBUTE_PEN_SIZE = 5;
int CC_ATTRIBUTE_FONT_ITALICIZED = 6;
int CC_ATTRIBUTE_FONT_UNDERLINE = 7;
int CC_ATTRIBUTE_WINDOW_FILL_COLOR = 8;
int CC_ATTRIBUTE_WINDOW_FILL_OPACITY = 9;
int CC_ATTRIBUTE_WINDOW_BORDER_TYPE = 10;
int CC_ATTRIBUTE_WINDOW_BORDER_COLOR = 11;
int CC_OPACITY_SOLID = 0;
int CC_OPACITY_FLASH = 1;
int CC_OPACITY_TRANSLUCENT = 2;
```

```
int CC_OPACITY_TRANSPARENT = 3;
int CC_BORDER_NONE = 0;
int CC_BORDER_RAISED = 1;
int CC_BORDER_DEPRESSED = 2;
int CC_BORDER_UNIFORM = 3;
int CC_BORDER_SHADOW_LEFT = 4;
int CC_BORDER_SHADOW_RIGHT = 5;
int CC_TYPE_ANALOG = 0;
int CC_TYPE_DIGITAL = 1;
int CC_PEN_SIZE_SMALL = 0;
int CC_PEN_SIZE_STANDARD = 1;
int CC_PEN_SIZE_LARGE = 2;
```

In org.ocap.media.ClosedCaptioningControl:

```
int CC_ANALOG_SERVICE_CC1 = 1000;
int CC_ANALOG_SERVICE_CC2 = 1001;
int CC_ANALOG_SERVICE_CC3 = 1002;
int CC_ANALOG_SERVICE_CC4 = 1003;
int CC_ANALOG_SERVICE_T1 = 1004;
int CC_ANALOG_SERVICE_T2 = 1005;
int CC_ANALOG_SERVICE_T3 = 1006;
int CC_ANALOG_SERVICE_T4 = 1007;
int CC_TURN_OFF = 0;
int CC_TURN_ON = 1;
int CC_TURN_ON_MUTE = 2;
```

In org.ocap.media.ClosedCaptioningEvent:

```
int EVENTID_CLOSED_CAPTIONING_ON = 0;
int EVENTID_CLOSED_CAPTIONING_OFF = 1;
int EVENTID_CLOSED_CAPTIONING_ON_MUTE = 2;
int EVENTID_CLOSED_CAPTIONING_SELECT_NEW_SERVICE = 3;
```

In org.ocap.media.AlternativeMediaPresentationReason:

```
int NO_ENTITLEMENT = 0x01;
int COMMERCIAL_DIALOG = 0x02;
int RATING_PROBLEM = 0x04;
int CA_UNKNOWN= 0x08;
int BROADCAST_INCONSISTENCY= 0x10;
int HARDWARE_RESOURCE_NOT_AVAILABLE= 0x20;
```

In org.ocap.media.MediaAccessAuthorization:

```
int UNLIMITED_VALIDITY_PERIOD= -1;
```

In org.ocap.media.MediaTimerListener:

```
int TIMER_START = 1;
int TIMER_STOP = 2;
int TIMER_WENTOFF_FIRST = 3;
int TIMER_WENTOFF_LAST = 4;
```

In org.ocap.system.EASModuleRegistrar:

```
int EAS_ATTRIBUTE_FONT_COLOR = 1
int EAS_ATTRIBUTE_FONT_STYLE = 2
int EAS_ATTRIBUTE_FONT_FACE = 3
int EAS_ATTRIBUTE_FONT_SIZE = 4
int EAS_ATTRIBUTE_BACK_COLOR = 5
int EAS_ATTRIBUTE_FONT_OPACITY = 6
int EAS_ATTRIBUTE_BACK_OPACITY = 7
```

In org.ocap.media.VBIFilterEvent:

```
int EVENT_CODE_FIRST_VBI_DATA_AVAILABLE = 1;
int EVENT_CODE_FORCIBLE_TERMINATED = 2;
int EVENT_CODE_VIDEO_SOURCE_CHANGED = 3;
int EVENT_CODE_FAILED_TO_DESCRAMBLE = 4;
int EVENT_CODE_TIMEOUT = 5;
int EVENT_CODE_BUFFER_FULL = 6;
int EVENT_CODE_TIME_NOTIFICATION = 7;
int EVENT_CODE_UNITS_NOTIFICATION = 8;
int VBI_DATA_FORMAT_XDS = 1;
int VBI_DATA_FORMAT_T1 = 2;
int VBI_DATA_FORMAT_T2 = 3;
```

```
int VBI_DATA_FORMAT_T3 = 4;
int VBI_DATA_FORMAT_T4 = 5;
int VBI_DATA_FORMAT_EIA608B = 6;
int VBI_DATA_FORMAT_NABTS = 7;
int VBI_DATA_FORMAT_AMOL_I = 8;
int VBI_DATA_FORMAT_AMOL_II = 9;
int VBI_DATA_FORMAT_UNKNOWN = 10;
int FIELD_1 = 1;
int FIELD_2 = 2;
int FIELD_MIXED = 3;
```

In org.ocap.hardware.VideoOutputPort:

```
int AV_OUTPUT_PORT_TYPE_RF = 0
int AV_OUTPUT_PORT_TYPE_BB = 1
int AV_OUTPUT_PORT_TYPE_SVIDEO = 2
int AV_OUTPUT_PORT_TYPE_1394 = 3
int AV_OUTPUT_PORT_TYPE_DVI = 4
int AV_OUTPUT_PORT_TYPE_COMPONENT_VIDEO = 5
int CAPABILITY_TYPE_DTCP = 0
int CAPABILITY_TYPE_HDCP = 1
int CAPABILITY_TYPE_RESOLUTION_RESTRICTION = 2
```

In org.ocap.test.OCAPTest:

```
int MAX_MESSAGE_LENGTH = 1500
byte MESSAGE_TERMINATION_BYTE = '\0'
int UDP = 0;
int TCP = 1;
```

In org.ocap.diagnostics:

```
int SNMP_TYPE_INVALID_OR_UNKNOWN = 0;
int SNMP_TYPE_INTEGER = 0x02;
int SNMP_TYPE_BITS = 0x03;
int SNMP_TYPE_OCTETSTRING = 0x04;
int SNMP_TYPE_OBJECTID = 0x06;
int SNMP_TYPE_IPADDRESS = 0x40;
int SNMP_TYPE_COUNTER32 = 0x41;
int int SNMP_TYPE_GAUGE32 = 0x42;
int SNMP_TYPE_TIMETICKS = 0x43;
int SNMP_TYPE_OPAQUE = 0x44;
int SNMP_TYPE_COUNTER64 = 0x46;
int MIB_ACCESS_READONLY = 0;
int MIB_ACCESS_READWRITE = 1;
int MIP_ACCESS_WRITEONLY = 2;
int SNMP_CHECK_FOR_SET_REQUEST = 0;
int SNMP_SET_REQUEST = 1;
int SNMP_GET_REQUEST = 2;
int SNMP_GET_NEXT_REQUEST=4;
int SNMP_REQUEST_SUCCESS=0;
int SNMP_REQUEST_NO_SUCH_NAME = 1;
int SNMP_REQUEST_SET_OUT_OF_RANGE = 2;
int SNMP_REQUEST_GENERIC_ERROR = 3;
```

# 19 RESOURCE MANAGEMENT

This section contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

In OCAP 1.1, as in DVB-MHP 1.0, the resources available in the integrated receiver decoder are shared between multiple applications. There are two major differences between OCAP 1.1 and DVB-MHP 1.0 that cause resource management to be handled slightly differently in the two environments:

- **In [DVB-MHP1.1.2], all user-level applications are service-bound.** Therefore, the network operator knows the maximum set of applications that are to execute concurrently and can ensure that they cooperate with one another concerning the use of the resources. **OCAP 1.1, however, allows for both service-bound and unbounded applications.** In OCAP 1.1, it is possible that applications can be obtained from sources external to the network operator and it is also possible for executables to be stored locally. Therefore, even if the network operator is aware in advance of all of the applications that the user may wish to execute, they typically do not know which set of applications a user wishes to execute concurrently, and, it is unlikely that they would be sure that there are sufficient resources for any arbitrary combination of applications. Additionally, if a particular MSO desires that applications that were not specifically authored for television in general, and OCAP 1.1 in particular, be executed, it is unlikely that the applications would cooperate with one another concerning the sharing of resources.
- **In the OCAP 1.1 environment, network operators are empowered to maintain control through the use of a network operator-specific Monitor Application.** When resources are over-subscribed in the DVB MHP 1.0 environment, and the offending applications are written using the [DAVIC] resource sharing framework (described in Annex F of [DAVIC]), the application manager MAY permit them to negotiate for the limiting resource. Failing a successful negotiation, the MHP application manager is the ultimate authority. OCAP 1.1 prefers to vest that authority in the Monitor Application rather than the application manager. Note that resource over-subscription is still possible in [DVB-MHP1.1.2] because (1) not all applications are written at the user-level, (2) applications bound to the same service are not required to be cooperative, and (3) hardware resources can fail, reducing the capabilities of a given integrated receiver decoder.

## 19.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection does not correspond to any sections or annexes of the [DVB-MHP1.1.2].

***Table 19–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| 19 Resource Management | No Corresponding Section | OCAP-Specific Extension |

## 19.2 OCAP 1.1 Specific Requirements

This requirement extends the specification requirements about resource management made to the [DVB-MHP1.1.2].

### 19.2.1 Normative

The following sections are normative for the OCAP 1.1 Profile.

#### *19.2.1.1 Resource Management in OCAP 1.1*

The [DVB-GEM 1.1] has a resource management system using the DAVIC Resource Notification API specified in Annex F of the [DAVIC] with an additional resource management rule, as described in Section 11.7.5 of [DVB-MHP1.1.2]. The OCAP 1.1 Profile supports the packages defined in Section 13.4.7 and Section 13.4.8, with extensions described in Section 19.2.1.1.1 and Section 19.2.1.1.5, below. That is to say, the following resource reserving methods SHALL obey the four steps defined below:

- org.davic.mpeg.sections.SectionFilterGroup.attach(TransportStream, ResourceClient, Object)
- org.davic.net.tuning.NetworkInterfaceController.reserve(NetworkInterface, Object)
- org.davic.net.tuning.NetworkInterfaceController.reserveFor(Locator, Object)
- org.havi.ui.HBackgroundDevice.reserveDevice(ResourceClient)
- org.havi.ui.HGraphicsDevice.reserveDevice(ResourceClient)
- org.havi.ui.HVideoDevice.reserveDevice(ResourceClient)
- org.ocap.media.VBIFilterGroup.attach(ServiceContext serviceContext, ResourceClient client, Object requestData)

Note that the org.dvb.event.RepositoryDescriptor class and its subclasses are out of the scope of this process. So the possible resource types specified as the resourceProxy parameter in the ResourceContentionManager.setResourceFilter() method and the ResourceContentionHandler.resolveResourceContention() method are following:

- org.davic.mpeg.sections.SectionFilterGroup
- org.davic.net.tuning.NetworkInterfaceController
- org.havi.ui.HBackgroundDevice
- org.havi.ui.HGraphicsDevice
- org.havi.ui.HVideoDevice

##### *19.2.1.1.1 Unconditional Rejection*

When a resource is going to be reserved by an application (directly or indirectly), first, an OCAP implementation SHALL check whether or not the application is allowed to reserve the resource by the Monitor Application. For this purpose, an application that has a MonitorAppPermission("handler.resource") permission can implement a subclass of org.dvb.application.AppsDatabaseFilter and register it to org.ocap.resource.ResourceContentionManager. The AppsDatabaseFilter.accept(AppID) method returns true if an application specified by the AppID is allowed to reserve the resource, or returns false if the application is not allowed to reserve it. A resource management system in the OCAP implementation calls this method and rejects the resource reservation if false is returned.

The AppsDatabaseFilter is set via the ResourceContentionManager.setResourceFilter(AppsDatabaseFilter, String) method. The implementation SHALL associate the AppsDatabaseFilter with the corresponding resource managing class (such as a determinate class of a ResourceProxy) in the OCAP implementation.

If an AppsDatabaseFilter has not been associated with the resource by the Monitor Application, then any application is allowed to reserve the resource.

##### *19.2.1.1.2 Negotiation*

If the application is allowed to reserve the resource, the OCAP implementation SHALL assess the availability of the resource. If there is no available resource that satisfies the reservation request, a resource contention occurs and negotiation begins. The OCAP implementation SHALL perform resource negotiation by calling the

ResourceClient.requestRelease(ResourceProxy, Object) method of every current owner of the resource in ascending order of application priority, from lowest to highest, until the resource becomes available.

#### *19.2.1.1.3 Resolving Resource Contention by Monitor Application*

If the resource request is still not satisfied after the negotiation phase, the Monitor Application MAY decide which application can reserve the resource.

For this purpose, an application that has a MonitorAppPermission("handler.resource") permission MAY implement a determinate class of org.ocap.resource.ResourceContentionHandler and register it to org.ocap.resource.ResourceContentionManager.

The implementation SHALL call, the ResourceContentionHandler.resolveResourceContention(ResourceUsage, ResourceUsage[]) method if resources could not be reserved through the DAVIC resource negotiation process.

If one or more resources, which are required to be reserved for the successful completion of a single call to a method in the OCAP API, cannot be acquired through DAVIC resource negotiations, the implementation SHALL create a single instance of a class implementing the ResourceUsage interface to represent resources required for the method call (for example ServiceContext.select() may require the implicit reservation of NetworkInterfaceController and HVideoDevice). The method ResourceUsage.getResourceNames() shall return fully qualified java class names for all resources that are required. The method ResourceUsage.getResource(String resourceName) may be used by the Monitor Application to retrieve resources already reserved by the implementation for the completion of the OCAP method call. If an application directly called a reserving method of a DAVIC resource, the implementation SHALL create an instance of the ResourceUsage interface to represent such a resource.

The implementation shall call the ResourceContentionHandler.resolveResourceContention(ResourceUsage, ResourceUsage[]) method with this newly created instance of ResourceUsage as the first parameter. The second parameter shall be an array of instances of ResourceUsage that represent all conflicting resource reservations. The method ResourceUsage.getAppID() for each entry in the array shall return the AppID of the application that has reserved the resources represented by the entry. The method ResourceUsage.getResources() shall return the fully qualified java class names for all the resources that are represented by that entry. The array of ResourceUsages returned by the method resolveResourceContention() specify the priority sequence in which applications are allowed to reserve the resources. The implementation SHALL reserve the resources included in the ResourceUsage according to this priority sequence and this may result in a current owner losing access to the resource in the case that the requester is earlier in the returned priority sequence. If a zero-length array is returned from the ResourceContentionHandler.resolveResourceContention(ResourceUsage, ResourceUsage[]) method, no application can reserve the resource (the current owner loses access to the resource).

**Note:** This priority sequence does not affect the application's priority attribute. It is used only for resolving the resource contention.

Upon successful resource reservation either implicitly by the implementation or explicitly by the application, the implementation shall maintain an instance of ResourceUsage corresponding to the resources reserved. This instance of ResourceUsage shall be used in creating the array of ResourceUsages used as the second parameter to resolveResourceContention() method in the event of any future resource contention involving a resource that is represented in the ResourceUsage. Implementation shall maintain this instance of ResourceUsage till all the resources are released implicitly or explicitly by the application.

#### *19.2.1.1.4 Resolving Resource Contention by OCAP Implementation*

If the ResourceContentionHandler.resolveResourceContention(org.ocap.resource.ResourceUsage newRequest, org.ocap.resource.ResourceUsage[] currentReservations) method returns null (i.e., ResourceContentionHandler has no instance) or a ResourceContentionHandler has not been associated with this resource, the inter application resource management process uses the application priority attribute (defined in Section 10.2.2.6) and the resource management process defined for each individual resource to resolve the resource contention. As part of the

resource management process, the implementation SHALL inform an application when a resource associated with the DAVIC resource API is removed by calling org.davic.resources.ResourceClient.release(ResourceProxy[]).

#### *19.2.1.1.5 Implicit Resource Reservation*

##### 19.2.1.1.5.1 HVideoDevice Reservation

When the implementation reserves a resource implicitly, the owner of the resource SHALL be the application that called the method that required the resource to be reserved implicitly. For example if the org.havi.ui.HvideoDevice.reserveDevice(ResourceClient) method is called by the OCAP implementation (possibly JMF Player) indirectly, the resource requester SHALL be an application that calls the org.havi.ui.HvideoDevice.getVideoController(), javax.tv.service.selection.ServiceContext.getServiceContentHandlers() or the overloaded javax.tv.service.selection.ServiceContext.select() method. If the org.davic.net.tuning.NetworkInterfaceController.reserve() or reserveFor() methods are called by the OCAP implementation indirectly, the resource requester SHALL be an application that calls the overloaded javax.tv.service.selection.ServiceContext.select() method. If any resources were reserved by the implementation before the Monitor Application was launched or if any resources were reserved by the implementation without explicitly or implicitly requested by OCAP applications, the method getAppID() for the ResourceUsage corresponding to the resource allocation should return null.

##### 19.2.1.1.5.2 NetworkInterface Reservation

When an application calls the javax.tv.service.selection.ServiceContext.select() method on a ServiceContext instance it has permission to present services with and a tune is required to complete the selection, the implementation SHALL attempt to reserve an org.davic.net.tuning.NetworkInterface instance for the application. The following behavior applies when resource contention occurs for such a reservation attempt:

- When a resource contention handler is registered; for any applications the implementation has reserved the NetworkInterface resource for, or is trying to reserve the resource for, the implementation SHALL use the application identifier corresponding to those applications when creating a ResourceUsage and calling the org.ocap.resource.ResourceContentionHandler.resolveResourceContention() method. If the application the resource is reserved for is destroyed after a successful implicit resource reservation, the implementation SHALL continue to use the application identifier of the application until another application reserves the resource. The implementation SHALL maintain the reservation for the destroyed application until the registered ResourceContentionHandler grants the resource to a different application.
- When a resource contention handler is not registered; for any applications the implementation has reserved the resource for or is trying to reserve the resource for the implementation SHALL use application priority corresponding to those applications to determine application priority for contention resolution. If the application the resource is reserved for is destroyed after a successful implicit resource reservation, the implementation SHALL continue to reserve the resource for that application until another application attempts to reserve the resource. For purposes of resource contention, in this case, the destroyed application SHALL be given an effective priority of zero.
- If the NetworkInterface reservation fails for a ServiceContext.select() method call, the select fails and the appropriate event is generated.

#### *19.2.1.1.6 Resource Management between Environments*

The rules and policies of the selected environment shall apply to resource conflicts between the selected environment and all other running environments. They do not apply to resources of which the selected environment is ignorant or to resource conflicts not involving the selected environment.

The rules and policies of a non-selected environment can still apply to applications from that environment, e.g., the lifecycle of OCAP applications can still be under the control of the monitor application and XAIT update monitoring continues even if the OCAP environment isn't selected.

Running applications whose environments are not selected are not excluded from using resources that are shared with the selected environment. However, they may not fully participate in the selected environment's resource negotiation process and hence must be prepared for resource requests to fail and for any shared resources they obtain to be removed at any time. The exceptions to this are resources which can be shared without participating in a negotiation process, e.g., graphics pixels (without changing the graphics resolution), platform exclusive user input events and user input events available to the application with focus.

If running applications whose environments are not selected hold resources not shared with the selected environment, then the rules by which they may lose those resources are outside the scope of this document.

Change of availability of shared resources shall always be reported to those OCAP applications which have registered for the appropriate ResourceStatusEvents. Specifically this includes the following:

- changes that happen as part of a change of selected environment
- changes due to the activities of non-cable applications
  NOTE: For avoidance of doubt, this does not mean that the DAVIC resource management mechanisms need to be used by non-cable applications.

#### *19.2.1.1.7 Tuners*

This section describes management of tuners. The term 'tuner' is used here refer to an entire 'video signal path', which is the combination of hardware and software resources required by a device to acquire and present video content. Some iDCRs may have cable and terrestrial tuners that share elements of the video signal path and hence receiving cable signals will block the reception of terrestrial signals and vice-versa.

Considered here is the non-DVR scenario. Note that reserving a tuner requires that the entire video signal path be configured for use by the environment for which it is reserved. Reservation of tuners on an iDCR specifically entails the following:

- When cable is the selected environment, an OCAP application may reserve a tuner, as modeled by the NetworkInterfaceController (NIC) object in Java, and the NIC will be of type cable.
- When cable is not the selected environment, an OCAP application may attempt to reserve a NIC of type cable. The iDCR will determine if the reservation is successful, i.e., the video signal path is configured for cable. If the iDCR is currently decoding a terrestrial signal, the reservation will fail if the cable and terrestrial signal paths have elements in common for any cable tuner whose signal path has elements in common with that of the tuner used to receive terrestrial signals.
- On transition from cable as the selected environment to another state, a reservation of a NIC may be lost, at discretion of the newly selected environment.
- On transition from non-cable to cable as the selected environment, OCAP applications may reserve a cable NIC. This will cause interruption of reception of the terrestrial signal via any terrestrial tuner whose signal path has elements in common with that of the newly reserved cable NIC.

### *19.2.1.2 Shared and non-Shared Resources*

Resources that are shared include the following where they are visible to both OCAP and non-OCAP environments:

- Application exclusive key events
- The right to control the resolution of graphics planes
- Tuners which can receive content from the cable network
- MPEG-2 section filters
- VBI filters

The following are not considered to be shared resources:

- Platform exclusive key events
- Receiving key events when an application has focus
- The right to control the resolution of graphics planes that are not used by cable applications
- Drawing pixels to graphics planes regardless of which other applications write to those graphics planes
- Tuners that cannot receive content from the cable network or which are not visible to the OCAP environment and cannot receive content from the cable network.
- MPEG-2 section filters which are never visible to OCAP applications, (e.g., those used by the implementation for access to MPEG PSI, DSMCC object carousel, AIT, ....)
- VBI filters which are never visible to OCAP applications.
- OOB section filters (regardless of whether they are used explicitly by applications (via the section filter API) or implicitly via java.net or the object carousel API.) These resources are exclusively used by the cable environment.
- OOB / DSG receivers and transmitters. These resources are exclusively used by the cable environment.

# 20 BASELINE FUNCTIONALITY

This section contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

This section of the OCAP 1.1 Profile specifies core cable network support functionality including support before a CableCARD device is inserted into an OCAP 1.1-compliant set-top box. The baseline functionality is part of the OCAP 1.1 implementation. Some of the baseline functionality SHALL be available to the viewer in the following cases:

- when a CableCARD device is not inserted
- before a Monitor Application is on-board

## 20.1 DVB-GEM and DVB-MHP Specification Correspondence

This section does not correspond to any sections or annexes of the [DVB-MHP1.1.2].

***Table 20–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| 20 Baseline Functionality | No Corresponding Section | OCAP-Specific Extension |

## 20.2 OCAP 1.1 Specific Requirements

This information extends the specification requirements made to [DVB-MHP1.1.2].

### 20.2.1 Conceptual Module (Informative)

Baseline Functionality can be thought of as sets of related functionality that can be grouped into conceptual modules as shown in Figure 20–1:

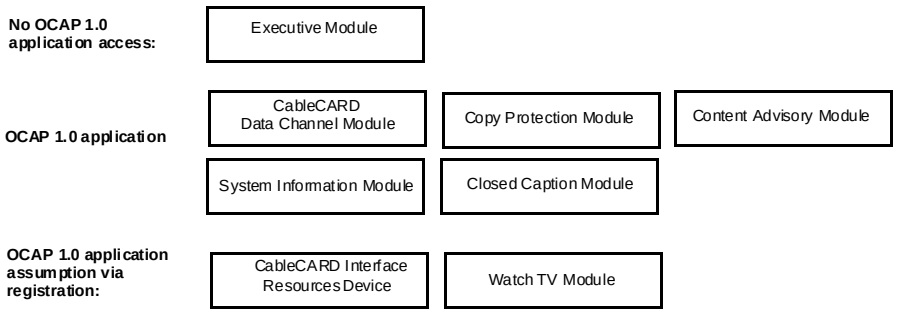



***Figure 20–1 - Baseline conceptual modules***

See normative sections in this chapter for module definitions and behavior.

### 20.2.2 Boot Process

This subsection details the boot process for the OCAP 1.1 implementation.

The boot process distinguishes between reboot scenarios, which begins at Step 1 in the subsections below, and initialization of the OCAP environment, which begins at Step 3 in Section 20.2.2.1 or Step 4 of Section 20.2.2.2. A CableCARD device may be inserted or removed at any time during the boot process. See Section 20.2.3, item 1 for requirements.

#### *20.2.2.1 Boot Process - CableCARD device absent*

1. **Power Applied or Reboot** - the set-top terminal has been powered on or rebooted via software.
2. **Hardware and Operating System Initialization** - low level initialization of the hardware and operating system
3. **Optional: Initialization of OCAP environment** - all of the system modules and components of the OCAP environment are initialized and started.
4. **Configure Environments** – The implementation SHALL place exactly one application environment in the selected state. The OCAP environment MAY be placed in any of the possible environments states.
5. **Optional: Launch Auto-start Unbound Applications -** For each abstract service in the services list that is signaled as "auto_select" by the Host Device Manufacturer, create a service context and select the abstract service in that service context.
6. **Begin Normal Operation** – see Section 20.2.3 below.

#### *20.2.2.2 Boot Process - CableCARD device present*

1. **Power Applied or Reboot** - the set-top terminal has been powered on or rebooted via software.
2. **Hardware and Operating System Initialization** - low level initialization of the hardware and operating system
3. Optional: Manufacturer Configuration - On first-time power up, manufacturer MAY select a non-cable environment and launch applications in that environment to allow customers to perform configuration operations. This step SHALL be followed by Step 4 below.
4. **Initialization of OCAP environment** - all of the system modules and components of the OCAP environment are initialized and started.
5. **CableCARD/Host Interface Initialization** - low level initialization of the CableCARD/Host interface.
6. **Code Download** - If there is a more recent release of OCAP 1.1 Implementation available on the network, upgrade the implementation to it as per Section 20.2.3.7, Download Module. The set-top terminal SHALL be running the most current OCAP 1.1 implementation for that terminal carried on the network.
7. **Load and Parse the XAIT** - Acquire the XAIT and update the Applications Database with each application signaled in the XAIT. Update stored application information as signaled in the XAIT (Refer to Section 11.2.2.3 and Section 12.2.6).

   The implementation SHALL launch the Initial Monitor Application with the highest version unless an application_storage_descriptor is present. In that case the Initial Monitor Application with the highest launch order is launched."
8. **Configure Environments** – The implementation SHALL place exactly one application environment in the selected state. If this is a first-time boot process, and an XAIT has been detected, the cable environment SHALL be the selected environment, otherwise, the implementation MAY select the environment that was selected before the boot process began and the OCAP environment MAY be placed in any of the possible environments states. If an XAIT is not detected, the implementation SHALL skip Step 9 below, and MAY continue the boot process at Step 10 or Step 11.
9. **Launch the Initial Monitor Application** - The implementation SHALL launch the Initial Monitor Application with the highest version unless an application_storage_descriptor is present. In that case the Initial Monitor Application with the highest launch signaled by the XAIT SHALL be launched. When an Initial

MonitorApplication is launched the next step in this boot process SHALL be deferred until one of the following scenarios is completed:

- 5 seconds have elapsed since the time the Initial Monitor Applications contructor has been invoked, and a call to MonitorConfiguringSignal() has not been made.
- The MonitorConfiguredSignal() method is called by an application with appropriate permissions.

See Section 20.2.4.1.1 and Annex O for invocation details of the monitorConfiguringSignal and monitorConfiguredSignal methods. If no Initial Monitor Application is detected in the XAIT or in storage, the boot process SHALL continue at Step 10 below.

The Initial Monitor Application MAY launch applications and services before calling the monitorConfiguredSignal method.

10. **Launch Auto-start Unbound Applications** - For each abstract service in the service list that is signaled as "auto_select" including services containing Host Device Manufacturer applications, create a service context and select the abstract service in that service context. If the version of a XAIT signaled application in application storage matches that defined in the XAIT, then the stored version MAY be launched. Otherwise, the signaled version SHALL be launched.
11. **Begin Normal Operation** – see Section 20.2.3 below.

### 20.2.3 Normal Operation

At this point, the Host device has reached the normal operating state and user input is taken and processed according to the applications that have been loaded and started through the boot process. After this, there are two sets of asynchronous events that could cause the OCAP 1.1 implementation to re-enter the boot process:

1. **Insertion or removal of CableCARD device -** If a CableCARD device is inserted, the implementation SHALL begin the boot process at Section 20.2.2.2 Step 4. **If a CableCARD device is removed, the implementation SHALL begin the boot process at Section 20.2.2.1 Step 3.**
2. **The Host device or OCAP Application initiates a reboot -** The Monitor Application SHOULD set a reboot handler to be notified of the reboot. The Monitor Application SHOULD terminate all applications and clean up resources when the reboot is notified. If initiated by the Host, or by an OCAP application and cable is the selected environment, a full reboot cycle beginning at Step 1 of the appropriate subsection above SHALL occur. Invocation of the reboot() method when cable is not the selected environment MAY be treated in any of the following ways: ignored, cause a reboot, or cause a re-initialization of the OCAP environment

During normal operation, the implementation SHALL poll for the presence of an XAIT. If an XAIT is detected when previously there was none, the implementation SHALL re-initialize the OCAP environment, see Section 20.2.2.2, Step 4.

### 20.2.4 Conceptual Module Descriptions

The conceptual module names are not normative, but their behavior SHALL be part of any OCAP 1.1 baseline functionality implementation.

#### *20.2.4.1 Executive Module*

The Executive Module has three primary functions:

- Discover, load, authenticate, and launch the appropriate Monitor Application
- Load and authenticate unbound applications signaled as auto-start in the XAIT, if no Monitor Application is signaled, then launch applications signaled as auto-start in the XAIT

- Management of applications stored in permanent memory

#### *20.2.4.1.1 Monitor Application Launching*

During the boot process the implementation SHALL parse the XAIT for applications signaled as the Initial Monitor Application (i.e., with an application priority of 255 and signaled as auto-start). If more than one Initial Monitor Application is signaled with different application identifiers, the choice for Initial Monitor Application to launch is implementation-dependent. The implementation SHALL create an abstract service for the Initial Monitor Application which is launched.

The implementation SHALL assure the download and storage of the most recent version of the initial Monitor Application in the OCAP 1.1 terminal when it is rebooted. If the signaled initial Monitor Application matches a Monitor Application that was previously stored in permanent memory, the Executive Module SHALL launch the Monitor Application in permanent memory. Otherwise the Monitor Application MAY be launched from the network without waiting for the downloading and storage process to complete.

Since applications are not restarted when a new version is signaled,, it is the responsibility of the running Initial Monitor Application to identify version changes. e.g., via the AppSignalHandler.notifyXAITUpdate() or OcapAppAttributes.hasNewVersion methods. When a version change occurs, the Monitor Application SHOULD and take the necessary steps to upgrade itself by terminating itself.

If an Initial Monitor Application fails to launch because the XAIT is missing information or contains incorrectly formatted data, the files cannot be reached, the ADF (if present) does not match the transport, a download error occurs, or an authentication error is encountered, the implementation MAY ignore Initial Monitor Application XAIT entries until a new version of the XAIT is received. If after an XAIT is detected that signals an Initial Monitor Application, the implementation SHALL attempt to load and authenticate the Initial Monitor Application. If loading and authentication is successful, the implementation SHALL create a pristine operating environment for OCAP, as if the OCAP environment has been initialized except that the Initial Monitor Application remains loaded and authenticated.

If for any of the reasons below an the Initial Monitor Application is not running during normal operation and an XAIT is detected that signals an Initial Monitor Application, the implementation SHALL attempt to load and authenticate the Initial Monitor Application. If loading and authentication is successful, the implementation SHALL create a pristine operating environment for OCAP, as if the OCAP environment has been initialized except that the Initial Monitor Application remains loaded and authenticated. Once this is accomplished the boot process SHALL be entered at Step 3. The implementation SHALL go through this process any time the Initial Monitor Application is re-launched. The circumstances that may lead to an Initial Monitor Application not running during normal operation include:

- The Initial Monitor Application failed to launch previously for any reason.
- The Initial Monitor Application was not included in an XAIT, or an XAIT was not present received before or during the boot process.
- The Initial Monitor Application is destroyed.

#### *20.2.4.1.2 Unbound Application Loading and Launching*

The implementation SHALL update the Applications Database with details of all applications signaled in the XAIT that are associated with services as and when the services are selected. If an Initial Monitor Application has been launched, the implementation SHALL wait for this application to signal that it has set its application filters and resource handlers, via the MonitorConfiguringSignal() and MonitorConfiguredSIgnal() APIS, see Annex O.

During the boot process the implementation selects each abstract service signaled as "auto_select" in the XAIT and attempts to load and launch any unbound applications belonging to these services signaled as auto-start. As new abstract services are selected, the implementation manages the loading and launching of applications signaled as

auto-start for those services. The loading and launching of applications by the Executive Module follows the rules specified in Section 10. Applications that are rejected by the application filtering currently set by the Initial Monitor Application are neither loaded nor launched. If memory resources are limited, auto-start applications with the highest priority SHALL be loaded and launched first. The implementation SHALL authenticate any loaded unbound applications. Any application that fails authentication SHALL be unloaded and SHALL NOT be launched, thus making room for other applications.

When an unbound application is signaled in an XAIT as auto-start and the following conditions are true:

- the application is located in a transport location in an inband channel carousel,
- the same version of the application is not currently stored in persistent storage,
- the application meets auto-start launch criteria or the application is the Initial Monitor Application,

then the implementation SHALL take the following steps in order:

1. Reserve a NetworkInterface where the transport stream carrying the application carousel can be tuned to. The implementation SHALL attempt to find a NetworkInterface not in use, but if all NetworkInterface instances are reserved the implementation SHALL pick a NetworkInterface, take the NetworkInterface from the current reservation, and reserve it for the application download. This is an implementation specific process and if a resource contention handler is registered it SHALL NOT be called upon to resolve the contention.

2. Tune to the transport the carousel is delivered in.

3. Download all of the application files to cache. If the application has an ADF the implementation SHALL use it to determine which files are downloaded. If the application does not have an ADF all of the files in and below the application directory indicated by the transport protocol descriptor and application location descriptor SHALL be downloaded.

4. Release the tuner after all of the application files have been downloaded.

5. Authenticate and launch the application.

6. If the application was signaled for storage, follow the rules in Section 12.2.3.

The following implementation behaviors apply for files and directories placed into cache for an unbound application with a transport location in an inband channel carousel:

- File and directories SHALL be maintained in cache until the application is destroyed with the following exceptions:
  - A class file MAY be removed from cache once it is class loaded if the application has no references to it.
  - A file that is not a class file MAY be removed from cache if the application created a DSMCCObject for it and calls the unload method and if the application has no references to it. The same is true for a directory if it does not contain referenced files or class files that haven't been class loaded.
- Rules defined in sections 13.3.7.2 and 16.2.1.4. The term "storage" refers to storage in cache in this case.
- A DSMCCObject SHALL not be implicitly created in the loaded state for a file or directory.
- A DSMCCObject successfully created by an application for a file or directory SHALL be created in the loaded state.

#### *20.2.4.1.3 Management of Stored Applications*

The Executive Module is responsible for the management of stored applications signaled in the XAIT as described in Section 12.

#### *20.2.4.2 Closed Captioning Module*

The Closed Captioning Module presents the closed captioning text when requested by the consumer. Regardless of the existence of a Monitor Application, OCAP 1.1 host device SHALL support the closed captioning defined in Section 7.2.5 of [HOST 2.0].

#### *20.2.4.3 System Information Module*

The System Information Module parses in-band and out-of-band SI. Clear-to-air in-band SI SHALL be processed with or without a Monitor Application running. It SHALL be made available through the corresponding OCAP 1.1 APIs. Out-of-band SI SHALL be processed once the CableCARD module is inserted and initialized. Out-of-band SI that is compliant with [SCTE 65] and the XAIT SHALL be made available through the corresponding OCAP 1.1 APIs as well.

In-band and out-of-band emergency alert system messages that comply with [SCTE 18] SHALL be forwarded to the Emergency Alert System Module.

#### *20.2.4.4 CableCARD Interface Resources Module*

The CableCARD Interface Resources Module sends and receives APDUs (Application Protocol Data Units) to and from the CableCARD device. CableCARD Interface Resources are defined in [CCIF 2.0].

The Specific Application Support Resource of the CableCARD Interface Resources Module SHALL support multiple assumable Private Host Applications. The Man Machine Interface Resource and the Application Information Resource are assumable modules. See Section 20.2.4.11.

#### *20.2.4.5 Copy Protection Module*

The Copy Protection Module controls copying of analog or digital content, and the output of content with respect to CCI information. Content delivered via a cable network SHALL NOT be stored locally on OCAP 1.1 devices.

##### *20.2.4.5.1 General Information*

OCAP 1.1 implementations SHALL fully protect all content as required by the CableCARD Host Interface License Agreement ([CHILA]).

20.2.4.5.1.1 Content Protection Requirements

Copy protection SHALL be implemented in conformance with the OpenCable CableCARD Copy Protection System specification, ([CCCP 2.0]). Under normal operation, the Conditional Access (CA) System delivers the Copy Control Information (CCI) securely to the CableCARD device. The CableCARD device passes CCI to the Host through a secure authentication protocol. The Host uses the CCI to control copy creation, analog output copy control encoding, and to set copy control parameters on Host outputs such as a 1394 port.

The CCI is a single byte, 8-bit, field and is defined in [CCCP 2.0]. It contains information related to both the analog and digital video outputs of the Host. In the 8-bit CCI field the Analog Protection System (APS) bits apply to the analog outputs and the Encryption Mode Indicator (EMI) bits apply to the digital outputs. The Host uses the APS bits to control copy protection encoding of analog composite outputs as described in Table 12 of [EIA-708-B] part B. The EMI bits are supplied to any Host digital output ports including the 1394 source port for control of copies made from those outputs.

#### *20.2.4.6 Content Advisory Module*

The Content Advisory Module is concerned with VCHIP content advisory handling from the VBI. This module assures that this information is directed to other interested system modules. This module is not concerned with

content advisory information found in the SI as that information is delivered to the corresponding OCAP 1.1 APIs for access by applications.

The Content Advisory Module is implemented by the Host device and is not exposed to OCAP applications. Host requirements for this module are described in OpenCable Host Device Core Functional Requirements [HOST 2.0], Section 8.2.6.

#### *20.2.4.7 Download Module*

The download module determines if the OCAP 1.1 compliant Host Device has a firmware upgrade version that is available via cable download. The download module complies with [CCIF 2.0]:

#### *20.2.4.8 Watch TV Module*

The Watch TV Module allows the consumer to view clear-to-air channels prior to CableCARD device insertion or Monitor Application launch. Minimum Watch TV capabilities include channel up and down, as well as numerical channel number entry.

The Watch TV Module is an assumable system module. See Section 20.2.4.11.

#### *20.2.4.9 CableCARD Data Channel Module*

The CableCARD Data Channel Module is one of the baseline functionality conceptual models that process APDUs on the CableCARD Data Channel. An OCAP-J application MAY use this functionality to send and receive APDUs on the CableCARD Data Channel via the org.ocap.system API. See Section 20.2.4.11.

#### *20.2.4.10 Emergency Alert System (EAS) Module*

The EAS Module receives emergency alert messages from the System Information Module, as per [SCTE 18]. It plays audio or displays text as specified by the messages.

The EAS Module is assumable with regard to audio presentation of EAS messages.

#### *20.2.4.11 Assumable System Modules*

OCAP-J applications that have appropriate MonitorAppPermission permission can assume certain system modules. This function is provided to replace an original manufacturer UI representation of the resident system module with an MSO's UI. For example, the MMI Resource defined in [CCIF 2.0] has a MMI dialogue. The OCAP-J application can process MMI APDUs (and Application Information APDU) and represent a MMI dialogue on behalf of the resident MMI Resource.

The method of assuming a system module depends on the system module. Detailed mechanisms are described in the following sections. Generally, an OCAP-J application that assumes a CableCARD Resource can send and receive APDUs to and from the CableCARD device by registering to the org.ocap.system. SystemModuleRegistrar (see Annex Q). It can also hide a UI representation of the resident system module. One OCAP-J application can take a role of multiple assumable modules, and also one OCAP-J application can take a role of single assumable module. It depends on implementation of the OCAP-J application. The resident system module SHALL relinquish all of its resources whenever the OCAP-J assuming application requests them. See also Section 19.

The OCAP-J assuming application SHOULD have a very high priority so that it is not destroyed by the application manager and can reserve scarce resources. It should not pause or destroy itself. See Section 10.2.2.6 for application priority values.

The following modules are assumable modules:

- Private Host Application on SAS Resource of the CableCARD Interface
- MMI Resource and Application Information Resource of the CableCARD Interface
- Watch TV module
- EAS module

#### *20.2.4.11.1 Private Host Application on Specific Application Support Resource*

[CCIF 2.0] supports Private Host Applications. The Private Host Application is a logical entity on the Host and it communicates with a vendor-specific CableCARD application on the CableCARD device. The CableCARD device and the Specific Application Support (SAS) Resource establish sessions on the Data Channel, and the Private Host Application and the vendor-specific CableCARD application use them to communicate using APDUs. The Private Host Application has a unique Private Host Application ID so that the vendor-specific CableCARD application can identify it. The Private Host Application reserves a session associating it with the Private Host Application ID via the sas_connect_rqst() APDU. The Private Host Application can't share a session with another Private Host Application (i.e., one Private Host Application can reserve only one session). The Private Host Application can't close the session since there is no public APDU to signal a close command from a Host side. The session is closed only when the Host shuts down the power, or when the CableCARD device is removed.

The Private Host Application is an assumable module. An OCAP-J application with MonitorAppPermission("handler.podApplication") permission can assume the Private Host Application(s) by registering to the org.ocap.system.SystemModuleRegistrar. If a Private Host Application that has matching Private Host Application ID has been registered already, it is unregistered automatically. Note that the Private Host Application is a logical entity so that one OCAP-J application can assume multiple Private Host Applications. The OCAP-J application can use the org.ocap.system.SystemModule and the org.ocap.system.SystemModuleHandler to exchange APDUs with a vendor-specific CableCARD application. A pair of the SystemModule and the SystemModuleHandler is associated with a specific Private Host Application ID and a specific session number. The OCAP implementation delivers incoming APDUs to the appropriate OCAP-J application according to the session number. See the following sections for details of registration and APDU delivery.

Note that the SAS Resource is not an assumable module. It is a CableCARD Resource that manages transmission of APDUs between Private Host Applications and a CableCARD device.

It is allowed to implement a native resident Private Host Application, for example it is installed via the Common Download. In such cases, the native Private Host Application shall follow the registration and unregistration via the org.ocap.systm.SystemModuleRegistrar described in the following sections. Note that a native resident Private Host Application SHALL be implemented as described in Section 7.2.1.5.1.4.3 and Section 10.2.2.7 OCAP applications and Native Applications.

20.2.4.11.1.1 Registration

OCAP-J applications can assume one or more Private Host Application(s) by registering with the org.ocap.system.SystemModuleRegistrar via the SystemModuleRegistrar.registerSASHandler() method. This method registers the specified SystemModuleHandler associating it with the specified Private Host Application ID and a session number selected by the OCAP implementation. The OCAP implementation can select either a session number that has already been established for the SAS Resource or a session number that will be established later for the SAS Resource, and it SHALL send a sas_connect_rqst APDU to the CableCARD device using the selected session to establish a SAS connection. If a sas_connect_cnf APDU with sas_session_status=0x00 returns on the same session from the CableCARD device, the SAS connection is established successfully. See [CCIF 2.0] for more information on establishing the SAS connection. If successful, the OCAP implementation SHALL call the SystemModuleHandler.ready() method with a new SystemModule instance. The SystemModule is associated with the same session number as associated with the SystemModuleHandler. If the connection attempt fails, the OCAP implementation SHALL call the SystemModuleHandler.ready() method with a null parameter and unregister the SystemModuleHandler automatically. Only a pair of a SystemModule instance and a SystemModuleHandler

instance shall be associated with a Private Host Application ID. Note that the Private Host Application ID doesn't have a relationship to the AppID of an OCAP-J application.

The SystemModuleRegistrar SHALL register a Private Host Application unless all of the SAS Resource sessions are consumed. The maximum number of SAS Resource sessions is 32. Note that a session is consumed by each registration since there is no session closing protocol for SAS Resource. The total number of possible concurrent Private Host Applications is lowered by one after each unregistration (i.e., a used session of the SAS Resource is never reused for another Private Host Application until the Host shut down the power or the CableCARD device is removed).

If another SystemModuleHandler that has a matching Private Host Application ID has already been registered, it shall be unregistered according to the Section 20.2.4.11.1.3 and then the specified new SystemModuleHandler is registered.

Note that the OCAP implementation SHALL register a native resident Private Host Application automatically only if the Initial Monitor Application has not registered a SystemModuleHandler that has a matching Private Host Application ID prior to the time the Initial Monitor Application signals the implementation that it has configured itself (see monitorConfiguredSignal(), Section 20.2.2). This restriction prevents registration competition between the OCAP implementation and the Initial Monitor Application (i.e., the Initial Monitor Application has a higher priority to register a Private Host Application).

See also the description of the SystemModuleRegistrar.registerSASHandler() method in Annex Q.

#### 20.2.4.11.1.2 Communication with CableCARD Device

If registration of a SystemModuleHandler is successful, the OCAP-J application implementing the handler can use the SystemModule returned by the SystemModuleHandler.ready() method to send an APDU to the CableCARD device. The SystemModule.sendAPDU() method sends the specified APDU bytes to the CableCARD device according to [CCIF 2.0]. The session number associated to the SystemModule is specified in the SPDU containing the APDU. Note that it is the responsibility of the OCAP-J application to specify the transaction_number field correctly in APDUs. The OCAP-J application can identify its own transaction as follows: The OCAP implementation delivers APDUs according to the session number associated with the Private Host Application and the session is never reused until the Host's power is turned off or removal of the CableCARD device, so that the OCAP-J application can receive only its own APDUs. Even if several Private Host Applications use a same transaction_number, the OCAP-J application can receive only APDUs containing its own transaction.

If errors are detected while sending APDUs, the OCAP-J Private Host Application is notified via the SystemModuleHandler.sendAPDUFailed() method. The OCAP-J Private Host Application MAY attempt to re-send the APDU. The OCAP implementation SHALL not re-send it.

The SystemModuleHandler.receiveAPDU() method is used to notify the OCAP-J application of receipt of an APDU. The OCAP implementation SHALL call the receiveAPDU() method of the SystemModuleHandler associated with the session number of the returned APDU. The OCAP implementation MAY dispose of an APDU after the receiveAPDU() method call returns. The OCAP implementation SHALL store incoming APDUs until the receiveAPDU() method calls for previous APDUs return.

The possible apdu_tags and APDU byte data for SAS Resource are specified in [CCIF 2.0].

For more information, see the description of the SystemModule and the SystemModuleHandler in Annex Q.

#### 20.2.4.11.1.3 Unregistration

When an OCAP-J application calls the SystemModuleRegistrar.unregisterSASHandler() method, the specified Private Host Application is unregistered. The unregistration process is as follows:

The notifyUnregister() method of the specified SystemModuleHandler or the SystemModuleHandler corresponding to the specified Private Host Application ID is called to notify unregistration. The notified OCAP-J application shall cease all APDU activity on this handler and return from this method to allow deregistration to complete. The application SHALL at least finalize all transactions (i.e., all of the transaction for the pair of the SystemModule and the SystemModuleHandler shall be terminated) so that other Private Host Applications can handle the transaction_number correctly. (The transaction_id is assigned by a sender of the sas_data_av APDU. The previous transaction shall be terminated not to confuse the transaction_id.) An OCAP-J application can send and receive APDUs to terminate current processes until the notifyUnregister() method returns. The OCAP implementation SHALL call this method in the unregisterSASHandler() method with an individual thread to avoid blocking.

**Note:** The session is not closed even if the associated SystemModule and SystemModuleHandler is unregistered, since there is no public protocol of session closing for SAS Resource and Private Host Application.

The OCAP implementation SHALL register any native resident Private Host Applications automatically when a SystemModuleHandler that has a matching Private Host Application ID is unregistered. This requirement ensures that resident functionality that is necessary for the host device manufacturer application is not lost permanently.

For more information, see the description of the SystemModuleRegistrar.unregisterSASHandler() method in Annex Q.

#### *20.2.4.11.2Man Machine Interface Resource and Application Information Resource*

[CCIF 2.0] specifies the Man Machine Interface (MMI) Resource and the Application Information Resource. The MMI Resource provides functionality to present an MMI HTML page. The MMI Resource uses the Application Information Resource to retrieve HTML files.

##### 20.2.4.11.2.1 Registration

OCAP-J applications with the MonitorAppPermission("handler.podApplication") permission can assume these Resources. The MMI Resource always uses the Application Information Resource, so OCAP provides a single API to assume these Resources.

An OCAP-J application can assume both the MMI Resource and the Application Information Resource by registering with the org.ocap.system.SystemModuleRegistrar via the SystemModuleRegistrar.registerMMIHandler() method. This method registers the specified SystemModuleHandler associated with the session numbers of the resident MMI Resource and the resident Application Resource. After an OCAP-J applications call the registerMMIHandler() method, the OCAP implementation SHALL call the SystemModuleHandler.ready() method with a new SystemModule instance. The SystemModule is also associated with the session numbers of the resident MMI Resource and the resident Application Resource. Only one SystemModule and one SystemModuleHandler can be registered for the pair of the MMI and the Application Information Resource. When the registerMMIHandler() method is called, the resident MMI Resource and the resident Application Information Resource don't necessarily terminate since they SHALL keep the established sessions but they SHALL relinquish scarce resources to the assuming OCAP-J application if requested (see Section 19), and they SHALL finalize current transactions so that the assuming OCAP-J application can manage the dialog_number deterministically. Resident MMI dialogs SHALL NOT be presented after successful registration. The OCAP implementation SHALL send a close_mmi_cnf APDU to the CableCARD device to notify it that an MMI dialogue closing. If the unregisterMMIHandler() is called or if the OCAP-J application that called registerMMIHandler() method changes its state to Destroyed, the resident MMI Resource MAY present subsequent MMI dialogs.

##### 20.2.4.11.2.2 Communication with CableCARD Device

After the registration, the OCAP-J application can use the SystemModule returned by the SystemModuleHandler.ready() method to send an APDU to the CableCARD device. The SystemModule.sendAPDU() method sends the specified APDU bytes to the CableCARD device according to [CCIF

2.0]. The SystemModule.sendAPDU() method SHALL distinguish the session according to the specified APDU (i.e., if the specified APDU is a MMI APDU, it shall be sent on the session established by the resident MMI Resource). If the specified APDU is an Application Information APDU it shall be sent on the session established by the resident Application Information Resource. The session number SHALL be specified in the SPDU containing the APDU.

**Note:** It is the responsibility of the OCAP-J application to specify the dialog_number field in the APDU correctly.

If errors are detected while sending APDUs, the OCAP-J application is notified via the SystemModuleHandler.sendAPDUFailed() method. The OCAP-J application MAY attempt to re-send the APDU. The OCAP implementation SHALL not re-send it.

The SystemModuleHandler.receiveAPDU() method is used to notify the OCAP-J application of receiving an APDU. The OCAP implementation shall call the receiveAPDU() method for both the MMI APDU and the Application Information APDU. The OCAP implementation may dispose of an APDU byte after the receiveAPDU() method call returns. The OCAP implementation SHALL store incoming APDUs until the receiveAPDU() method call with a previous APDU return.

The possible apdu_tags and APDU byte data for the MMI Resource and the Application Information Resource are specified in [CCIF 2.0]. The OCAP implementation doesn't have to confirm the validity of the specified APDU parameter but it shall investigate the apdu_tag value to identify which session should be used.

See also the description of the SystemModule and the SystemModuleHandler in Annex Q.

20.2.4.11.2.3 Unregistration

If the SystemModuleRegistrar.unregisterMMIHandler() method is called, the SystemModuleHandler instance and the SystemModule instance registered via the registerMMIHandler() method are unregistered according to the following process. The notifyUnregister() method of the SystemModuleHandler is called to notify unregistration. The notified OCAP-J application shall cease all APDU activity on this handler and return from this method to allow deregistration to complete. At least all transactions shall be finalized and all MMI dialogs shall be closed so that the alternative MMI Resource and Application Information Resource can handle the dialog_number certainly. The OCAP-J application shall send the close_mmi_cnf APDU to the CableCARD device to notify MMI dialogue closing. Note that the OCAP-J application can send and receive APDU for a terminating process until notifyUnregister() method returns. The OCAP implementation shall call this method in the unregisterMMIHandler() method with an individual thread to avoid blocking. Note that the session is not closed.

See Annex Qfor more details.

### *20.2.4.11.3Watch TV Module*

The Watch TV Module is an assumable module. OCAP-J applications that have any MonitorAppPermission can become an assuming Watch TV Module. There is no special API to assume Watch TV Module.

The resident (assumed) Watch TV shall give all of its resources (i.e., key events and a tuner device, to any OCAP-J applications that have any MonitorAppPermission, whenever requested). Note that the resident Watch TV Module need not terminate while being assumed, but it shall not disturb a presentation of the assuming Watch TV application.

### *20.2.4.11.4EAS Module*

The EAS module is a conceptual module in the implementation that handles EAS messages in accordance with [SCTE 18]. The OCAP implementation shall exclusively reserve resources necessary for EAS handling. If applications had explicitly reserved these resources prior to the device entering an EAS state, the resources would be taken away using the DAVIC resource negotiation framework. If applications do not voluntarily release

resources through the DAVIC resource negotiations, resources would be unilaterally taken away. If resources implicitly reserved by the implementation are taken away for EAS handling, the activity for which the resources were used would be terminated due to unavailability of resources. The effect on applications when resources are taken away for EAS handling would be similar to the effect of resources being taken away by a higher priority request. The resource contention handler will not be invoked when resources are acquired for EAS Handling.

If the OSD output of applications would disrupt EAS message presentation to the user, the implementation should make the root containers of all applications invisible. Applications should be notified using the appropriate AWT event notification.

The implementation would make sure that the resources necessary for EAS handling cannot be acquired by applications while the EAS handling is in progress.

What resources are used for EAS handling is implementation dependant. The implementation is responsible for making sure that EAS handling conforms to SCTE 18 or other standard specifications and also to make sure that applications cannot disrupt EAS handling. For example, for EAS messages requiring forced tune, a tuner, a video decoder, a video plane and audio decoder should be acquired by the implementation. In addition, the implementation may have to acquire the graphics plane and any other video planes so that other applications cannot obstruct the EAS presentation.

An application can interact with the EAS Module using three distinct techniques:

3. Listen for EAS events and check EAS state via the org.ocap.system.EASManager;
4. Query and set EAS display attributes using the org.ocap.system.EASRegistrar;
5. Register to be notified of EAS audio descriptors and provide audio if none is specified by signaling as defined by [SCTE 18], also using the EASRegistrar.

The implementation SHALL generate EAS events. Applications can listen for these events using the EASManager class, EASListener interface, and EASEvent class defined in Annex Q. When an EAS message is received the implementation SHALL generate an EAS event with a reason code of EAS_DETAILS_CHANNEL or EAS_TEXT_DISPLAY depending upon the corresponding field values in the EAS table. Resources SHALL NOT be taken away from the application for EAS use until all registered listener warn methods have been called for the first event of an EAS message. The implementation can remove resources as soon as the method is called and is not required to wait for listeners to return from this method. This event SHALL be generated every 10 seconds during the EAS message until the message completes. At which time the implementation SHALL generate an EAS event with reason code EAS_COMPLETE. If an EAS message is in-progress when a listener registers the implementation SHALL NOT call the listener's warn method, but SHALL call its notify method immediately.

Parental control locks SHALL NOT block EAS presentation (it is the responsibility of the implementation to unblock any implementation specific parental control locks to enable EAS presentation. When EAS completes it is the responsibility of OCAP applications to restore its state including any service presentation. At the end of the EAS presentation implementation specific parental controls SHOULD be applied.

An OCAP-J application that has MonitorAppPermission("handler.eas") can assume EAS Module audio presentation.

[SCTE 18] specifies screen representation of an alert text in the alert_text() descriptor. The OCAP-J application can set and get a preferred attribute value of an alert text (e.g., font/background color and opacity, font style, font type face and font size, via the EASModuleRegistrar.setEASAttribute() and getEASAttribute() methods). The possible attribute value can be retrieved by the EASModuleRegistrar.getEASCapability() method.

The OCAP-J application can also set an EAShandler. If the alert_priority=15, but no audio specified by [SCTE 18] is available, the OCAP implementation shall call the EASHandler.notifyPrivateDescriptor() method. The OCAP-J application can get the location of an alternative audio resource specified in the private descriptor and play it

according to [SCTE 18] (This audio corresponds to the step of "Audio available for use with scroll?*", in Figure 1). The OCAP implementation notifies that the specified alert duration finished via the EASHandler.stopAudio() method. If the OCAP-J application doesn't support the private descriptor, the EAShandler.notifyPrivateDescriptor() method shall return false and the OCAP implementation can play detailed channel or proprietary audio.

See Annex Qfor more details.

# 21 MONITOR FUNCTIONALITY

This section contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

This section describes the OCAP 1.1 monitor functionality. The functionality described in this section MAY be implemented in a single Monitor Application or MAY be implemented as a set of applications in the same abstract service as, and under the control of, an initial Monitor Application. In either case the Monitor Application to be started first is identified by having an application priority of 255. Additionally it SHOULD be the only AUTO-START application in its abstract service. If more than one initial Monitor Application is signaled, the choice for which to launch is implementation-dependent.

## 21.1 DVB-GEM and DVB-MHP Specification Correspondence

This section does not correspond to any sections or annexes of the [DVB-MHP1.1.2] specification.

***Table 21–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| 21 Monitor Functionality | No Corresponding Section | OCAP-Specific Extension |

## 21.2 OCAP 1.1 Specific Requirements

### 21.2.1 Extensions to DVB-GEM (Informative)

#### *21.2.1.1 Overview*

The monitor functionality is provided by a set of special unbound Monitor Applications with access to privileged API sets. These applications MAY use these APIs to help manage the execution of OCAP-compliant applications in the OCAP Host device. The set of functionality provided by this application set is referred to as "the monitor" in the remainder of this section. The term "Monitor Application" in other parts of this specification is generally meant to refer to this set of functionality, except in cases where it explicitly refers to that application that is assigned the highest available application priority.

The monitor should include a single AUTO_START application in its own service (i.e., with no other AUTO_START applications in that same service with the same priority) with an application priority of 255. This AUTO_START application is known as the initial Monitor Application. The monitor MAY include a number of other applications which are launched in the same service or MAY provide all of the monitor functionality in the initial Monitor Application. The service containing the Monitor Application SHALL be signaled as "auto_select".

**Note:** It is not a requirement for the monitor functionality to be deployed, therefore, all of its specification is optional for the MSO. However, all implementations are required to provide all of the mandatory APIs defined in this specification including those that support monitor functionality.

The monitor functionality provides a number of specific capabilities:

- The monitor MAY register unbound applications with the applications database. (See Section 21.2.1.6.)
- The monitor MAY validate the starting of all applications through the setting of application filters. (See Section 21.2.1.7.)

- The monitor MAY be informed of changes in the XAIT signaling through the setting of an application signal handler. On receipt of a new version of the XAIT, the monitor MAY instruct the implementation to ignore this version of the XAIT. (See Section 21.2.1.8.)
- The monitor MAY change the set of permissions available to an application during the application life cycle. (See Section 21.2.1.9.)
- The monitor MAY communicate with a proprietary application on the CableCARD device using a specific application resource or MAY communicate with generic CableCARD resources. (See Section 21.2.1.10.)
- The monitor MAY ask the system to reboot, and register to receive an event when the system decides to reboot. (See Section 21.2.1.18.)
- The monitor MAY provide a Resource Contention Handler for managing resource contention deadlocks, and filter application requests for resources. (See Section 21.2.1.12.)
- The monitor MAY modify the priority of other applications. (See Section 21.2.1.13.)
- The monitor MAY filter User Input events, and change their value, and redirect them to specific applications. (See Section 21.2.1.14.)
- The Monitor Application MAY assume the role of the Emergency Alert System handler. (See Section 21.2.1.15.)
- The Monitor Application MAY set preference of Closed Captioning message representation. (See Section 21.2.1.16.)
- The Monitor Application MAY indicate to the OCAP implementation whether testing APIs should be initialized or not. (See Annex O.)
- The Monitor Application MAY initiate a code download. (See Section 21.2.1.19.5.)

The MSO is responsible for the development, maintenance, and delivery of the monitor. The monitor's complexity and feature set is a function of the MSO's requirements.

Figure 21–1 shows the relationship between the Monitor Application, the OCAP 1.1 Application Environment and other applications.

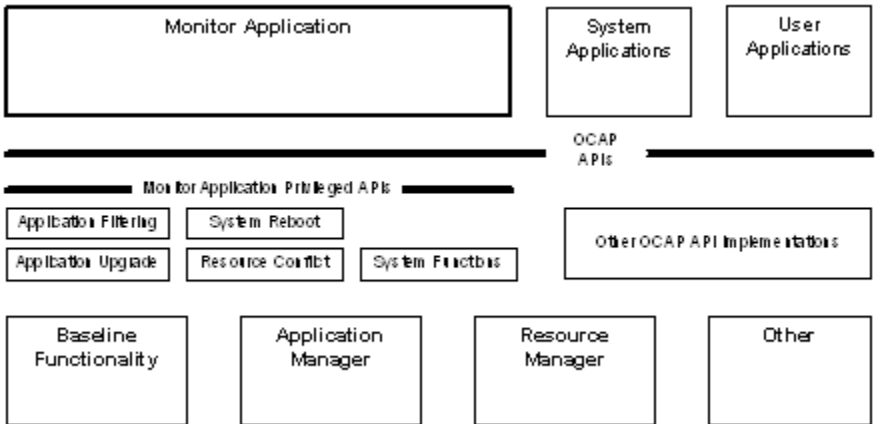



***Figure 21–1 -OCAP 1.1 Applications***

There SHOULD be only one initial Monitor Application signaled as AUTO_START in the XAIT. For varied functionality and to delegate capabilities, the initial Monitor Application can start multiple applications. For example, the initial Monitor Application may start another unbound application for input event handling.

During boot-up, the executive module creates an abstract service for the initial Monitor Application defined in the XAIT and then starts it according to the rules specified in Section 20.2.4.1.1.

It is the responsibility of the initial Monitor Application to manage upgrades to both itself and other monitor component applications. Management of upgrades can be accomplished by selectively allowing the applications database to be updated upon detection of a new version of the XAIT, see notifyXAITUpdate(), or by responding to updates to the applications database, detected by installing an AppsDataBaseEventListener (see [DVB-MHP1.1.2] Annex S). Ensuring compatible versions of monitor components may be important as the initial Monitor Application may decide to delay the installation of a monitor component in the case that this would affect viewer experience or in the case that component application expects other components to be concurrently installed and these cannot be downloaded.

Section 20.2.4.1.1 defines how the implementation is responsible for assuring the download and storage of the most recent version of the Monitor Application in the OCAP 1.1 terminal when it is rebooted. This will assure validity of the Monitor Application, when:

- A brand new OCAP 1.1 terminal has been installed for the first time
- The OCAP 1.1 terminal has been moved from one network to another
- A new version of the Monitor Application has been released

#### *21.2.1.2 Signaling*

The Monitor Application is signaled in the XAIT. The Monitor Application is signaled like any other application with the exception that the application_priority field in the application descriptor shall be set to the highest possible value of 255. The initial Monitor Application is signaled by setting the application_control_code field to AUTO_START.

#### *21.2.1.3 Launching*

Section 20.2.4.1.1 specifies how the Initial Monitor Application signaled in the XAIT is started by the implementation.

The Initial Monitor Application MAY launch other Monitor Applications within its own service to provide the full range of monitor functionality. The initial Monitor Application manages the life cycle of the applications that it launches.

After launching, the Initial Monitor Application MAY signal to the implementation that it is ready for other services to be started. This signal MAY be sent once all of the handlers and all of the application filters have been registered to instruct the implementation to proceed with the remainder of the boot procedure. See Section 20.2.2.2.

#### *21.2.1.4 Stopping*

The auto-started initial Monitor Application may voluntarily stop itself. In fatal error circumstances, the implementation may change the state of the initial Monitor Application.

#### *21.2.1.5 Authentication*

Section 14.2.2.2 specifies how applications may request Monitor Application permissions and the rules for deciding if these can be granted.

#### *21.2.1.6 Application Registration*

A Monitor Application that has MonitorAppPermission("registrar") can register and unregister applications in the Applications Database using the AppManagerProxy.registerUnboundApp() and the AppManagerProxy.unregisterUnboundApp() methods respectively. Applications are registered to a specific abstract

service and have the same life cycle semantics as applications signaled in the XAIT signaling. Only applications that are registered using the registerUnboundApp() method MAY be unregistered by unregisterUnboundApp().

#### 21.2.1.7 Application Filtering

A Monitor Application that has MonitorAppPermission("handler.appFilter") can set an application filter via the AppManagerProxy.setAppFilter() method. After the Monitor Application has indicated that it has set its filters, the accept() method of the filter in the Monitor Application is called prior to launching any application. If the method returns "false" for the application to be launched, the application is not launched; otherwise, the implementation continues with the process of launching the application.

The request to launch can be from any of service selection, application signaling or via the AppProxy.start() method. See Annex G for more details.

#### 21.2.1.8 Application Signal Handling

A Monitor Application that has MonitorAppPermission("handler.registrar") can set an application signal handler via the AppManagerProxy.setAppSignalHandler() method. This facility allows Monitor Applications to reject an updated XAIT in which case the applications database and stored applications are not changed. This facility is fully defined in the specification for org.ocap.application.AppSignalHandler.

The array of OcapAppAttributes provided to the notifyXAITUpdate() method includes the attributes associated with all applications whose details are listed in this XAIT. If this application is associated with a current service, the current set of attributes can be obtained from the org.ocap.service.AbstractService.getAppAttributes() method for the same AppID. The monitor can then compare the new set of attributes listed in the XAIT against the current set of attributes stored with the service and use this as a basis for the return value for the notifyXAITUpdate() method. The monitor can only identify applications that are no longer signaled in the new version of the XAIT by comparing the new list of attributes against all applications in each of the currently listed services.

#### 21.2.1.9 Security Policy Handling

A Monitor Application that has MonitorAppPermission("security") can set a security policy handler via the AppManagerProxy.setSecurityPolicyHandler() method. The OCAP implementation calls the getAppPermissions() method of the security policy handler whenever it launches any type of application. The implementation of this method provided by the Monitor Application can filter the set of permissions to be granted to the application when launched. See Annex G for more details.

#### 21.2.1.10 CableCARD Communications

The MSO may opt to set up a system module to provide proprietary communications between a Monitor Application with MonitorAppPermission("podApplication") permission and an application on the CableCARD device. This can be accomplished using the specific application support resource as accessible through the SystemModuleRegistrar API, see Annex Q, and Section 20.2.4.11.1.

#### 21.2.1.11 Hardware API

An application with MonitorAppPermission("setVideoPort") is allowed to enable or disable any of the video output ports on the terminal. See Annex F for more details.

#### 21.2.1.12 Resource Management

A Monitor Application that has MonitorAppPermission("handler.resource") can set a resource filter via the ResourceContentionManager.setResourceFilter () method and also set a resource contention handler via the ResourceContentionManager.setResourceContentionHandler () method. The resource filter and the resource

contention handler are used in a resource reservation process to modify a resource management policy according to the individual policy of the MSO. See Section 19 and Annex L for more details.

#### *21.2.1.13 Application Priority Management*

A Monitor Application that has MonitorAppPermission("handler.resource") permission can set the priority of any other application, except for applications with Monitor Application priority. The org.ocap.application.OcapAppAttributes interface contains a setPriority() method that provides this capability. If the application is in the Xlet Active state, or if the application has Monitor Application priority, the setPriority method will do nothing. The priority set by the setPriority method shall persist until a new version of the application is signaled, a reboot occurs, or the setPriority method is called again with a different priority value.

#### *21.2.1.14 User Event Filtering*

A Monitor Application that has MonitorAppPermission("filterUserEvents") can set a user event filter via the EventManager.setUserEventFilter () method and change the disposition of non-mandatory ordinary key codes via the UserEventFilter.FilterUserEvent() method. The change in disposition includes changes to the type of event or to the recipient of the event. See Annex K for more details.

#### *21.2.1.15 Emergency Alert System Configuration*

A Monitor Application that has MonitorAppPermission("handler.eas") can set preferred attributes of alert text and register a handler to retrieve a private descriptor in a cable_emergency_alert() of Emergency Alert System messages using the EASModuleRegistrar.setEASHandler() method. See Annex Q for more details.

#### *21.2.1.16 Closed Captioning Display*

A Monitor Application that has MonitorAppPermission("handler.closedCaptioning") can set preferred attribute values of closed captioning text representation (e.g., font/background pen color, font style and size etc. via the org.ocap.media.ClosedCaptioningAttribute class). An OCAP implementation uses the specified preferred value when drawing a closed captioning text on a screen. The application can also get capabilities of a closed captioning module in a host device. An analog captioning module may have a different capability from a digital captioning module. It is not required to support all kind of closed-captioning attribute. The ClosedCaptioningAttribute.getCCCapability() method returns supported values for each attribute.

The application can select a closed-captioning service number to be represented on a screen via org.ocap.media.ClosedCaptioningControl class. One of analog services (CC1 to CC4, T1 to T4) defined in [EIA-608-B] and digital services (Service #1 to #6) defined in [EIA-708-B] is available. And the application can also turn the selected captioning service on, off and on only when muting an audio. Note that at least one captioning service is required to be decoded at once on the OCAP implementation. Multiple captioning decoding is implementation dependent. See Annex S for more details.

#### *21.2.1.17 VBI Data Filtering*

A Monitor Application that has MonitorAppPermission("vbifiltering") can filter VBI data from VBI lines of an analog video and from an user_data structure in a MPEG picture header.

NTSC 525-line 60-field-per-second system has VBI lines that can deliver data bit sequence in analog video. A VBI line wave format consists of a common part and a proprietary data bit sequence part. The common part is defined in NTSC signal standard and consists of a horizontal sync and a color burst. The proprietary data bit sequence is defined in each VBI usage specification and consists of a bit sequence in a proprietary data rate. The bit sequence may be a simple character sequence, or may have a packet structure. Furthermore, the line number and interleaving techniques are also different. For example, [EIA-608-B] defines a waveform of line 21 for closed captioning. The captioning text is a non-packetized simple character sequence, but XDS is a packet with a start and end code. However [EIA 516] defines another wave format for NABTS teletext and uses line 10 to 20, and [SCTE 20] and

[SCTE 21] defines a transmission manner of line 21 VBI data in MPEG video stream. Such VBI data is extracted in a host device and inserted in a decoded analog video to deliver closed captioning data (also called reconstruction).

The org.ocap.media package provides VBI data filtering API to retrieve these data bit sequences. To support bit mask filtering of any type of data, OCAP specifies a "VBI data unit" for each VBI usage specification. For XDS, one XDS packet identified by a start and end code is a VBI data unit. So the VBI data unit for XDS is variable length. See [EIA-608-B], Section 9. For Text services (T1 to T4) of EIA/CEA 608-B [EIA-608-B], one data unit is bytes between a Text service in code and an out code. For a generic EIA/CEA 608-B [EIA-608-B] data, one data unit is a set of two characters that consist of Character One and Two. For NABTS, one NABTS "Data packet" that consists of a 5 bytes prefix and a 28 bytes data block with optional suffix is a VBI data unit. So the VBI data unit for NABTS is fixed 33 bytes length. See also [EIA 516] Section 2. For AMOL, a fixed length bit sequence in one VBI line is a VBI data unit. See Summary of AMOL I and AMOL II Specifications, Nielsen Media Research [59] for more details. And to filter unknown type data byte, OCAP defines an UNKNOWN type data format. For UNKNOWN, a bit sequence in one VBI line is a data unit. The number of bits and a bit rate of the UNKNOWN data format depend on VBI usage specification to be filtered. See the description of the VBIFilterGroup class for more details.

VBI data filtering is processed as follows: An application creates an org.ocap.media.VBIFilterGroup instance that contains a required number of VBI filters. The application can create VBI filters to retrieve VBI data of a specific data type in a specific VBI line and field via newVBIFilter() method. When VBIFilterGroup.attach() method is called, all VBI filters in the VBIFilterGroup are reserved according to Section 19. The startFiltering() method starts actual filtering of VBI data in a VBI line of currently selected video on the attached ServiceContext. A VBIFilter filters data units until a buffer is full or a stopFiltering() call, and provides a sequence of data units of a specified data format in arrival order. If the current video is an analog video, the VBI filter retrieves the specified data unit in the specified line and field. If the current video is a MPEG video and if a line 21 is specified, the VBI filter retrieves a line 21 data unit from a user_data in a MPEG picture header according to [SCTE 20] and [SCTE 21]. Filtering status is notified by a VBIFilterListener and a VBIFilterEvent.

The bit masking is applied to VBI data as follows. A mask bit (posFilterMask, negFilterMask) defines which bits are filtered on. The value defines a bit sequence that a filtered data unit should have. In case of positive masking, only when all bits in the data unit in the masked range are equal to the posFilterDef value (i.e., only the following situation occurs, the data unit will be retrieved):

posFilterDef and posFilterMask == single data unit and posFilterMask

In case of negative masking, only when bits in the data unit in the masked range are different from the negFilterDef value (i.e., only the following situation occurs, the data unit will be retrieved):

negFilterDef and negFilterMask != single data unit and negFilterMask

At a minimum, OCAP 1.1 SHALL support filtering of CC1, CC2, T1 and T2 data format in VBI line 21 field 1 and CC3, CC4, T3, T4 and XDS data format in VBI line 21 field 2 as specified in [EIA-608-B]. Note that filtering of these caption and text mode service data is not provided to display captioning text on a screen synchronizing with video and audio. It is not guaranteed that filtering performance satisfies expected delay.

Filtering of a combination of the other line, field and data format is optional. Support of [SCTE 20] and [SCTE 21] is optional.

#### *21.2.1.18 Reboot*

A Monitor Application with MonitorAppPermission("reboot") can call the Host.reboot() method to initiate a reboot of the Host device.

#### *21.2.1.19 System Event Handling*

Using org.ocap.event.system.SystemEventManager a trusted application with MonitorAppPermission("systemevent") can register to receive system events of type reboot, resource depletion, and error. A handler implements the org.ocap.event.system.SystemEventListener interface so that the implementation can notify the handler of new events. The implementation will pass an org.ocap.event.system.SystemEvent to the handler. Each of the event types mentioned SHALL be registered for separately. This interface allows the handler to determine the type of event, as well as other event related information. See Annex U.

##### *21.2.1.19.1 Reboot Handling*

When notified of an impending reboot, the registered system event listener can perform actions before returning from the notification method. The implementation SHALL wait for the Monitor Application to return before proceeding with the reboot. The implementation will specify the cause of the reboot when calling the handler (e.g., button pressed, hardware failure, Monitor Application, implementation). The implementation SHALL NOT allow the reboot process to lock up and SHALL time out and continue processing if the notifyEvent() method has not returned within an implementation specific time that SHALL be set in the range of 5 to 60 seconds.

##### *21.2.1.19.2 Resource Depletion Handling*

When the implementation has determined that it has insufficient resources to continue the execution of one of the running applications (as defined in [DVB-MHP1.1.2] Section 9.1.4.4, "Stopping by the MHP terminal due to a shortage of resources") and if a resource depletion handler is registered then the implementation SHALL call the handler's notify method (i.e., org.ocap.system.event.SystemEventListener.notifyEvent) in order to provide an opportunity for the handler to stop execution of one or more applications. The resource depletion handler SHALL NOT be called under other circumstances. The implementation SHALL use the resource depletion events specified in the org.ocap.system.event.ResourceDepletionEvent class and indicate the appropriate event when calling the notifyEvent() method. When a resource depletion handler's notifyEvent() method is called the implementation SHALL wait for the handler to return from the method before destroying any applications. The implementation SHALL NOT allow the resource recovery process to lock up and SHALL time out and continue processing if the notifyEvent() method has not returned within an implementation specific time that SHALL be set in the range of 5 to 60 seconds.

##### *21.2.1.19.3 Error Logging*

Errors and informational messages may be logged by the implementation or an application. An application can log an error or informational message by first getting the singleton event manager using the org.ocap.event.system.SystemEventManager.getInstance() method, then by calling the log method. If an application makes a method call that throws an exception, but is not caught by the application, the implementation SHALL pass the error to any registered system event listener.

Applications may generate error event types that are outside the OCAP definition. Applications SHOULD NOT generate error event types that overlap with OCAP reserved ranges for applications, otherwise the implementation may misinterpret them.

##### *21.2.1.19.4 Application Error Events*

Applications can only pass system-defined error events such as org.ocap.system.event ErrorEvent and RebootEvent types. Applications that create their own subclasses of SystemEvent cannot pass those events through the event system. (This is due to implementation and security issues.)

##### *21.2.1.19.5 Deferred Download Events*

When the implementation receives a CVT ([CCIF 2.0]) that signals a has a download_command value = 01 (deferred download ), the implementation SHALL generate a deferred download system event so that a Monitor

Application that has registered as the listener for this event can call org.hardware.Host.codeDownload() method. It is up to the application to determine the appropriate time to initiate the download.

#### *21.2.1.20 System Properties Permissions*

A java.util.PropertyPermission with action string "read" shall be granted for applications withMonitorAppPermission("properties") for all properties listed in Table 13–3:

The permission shall be denied for the action string "write". Applications without MonitorAppPermission("properties") shall be denied permission to read or write the properties in Table 13–3.

#### *21.2.1.21 Shared Classes*

A privileged application can be given the capability to share class files with other applications, remove shared class files, or access class files that have been shared by another application.

##### *21.2.1.21.1 Registered API*

The org.ocap.system.RegisteredApiManager class contains methods that let an application share classes with other applications, update shared classes, or remove shared class access from other applications; see Annex Q.

Application access to shared classes is controlled by RegisteredApiUserPermission(). The mechanisms and locations for enforcing these access controls are defined in the specification for org.ocap.system.RegisteredApiManager, see Annex Q.

**Note:** An application that desires to access shared classes need not be granted MonitorAppPermission(); furthermore, it need not be an unbound application. To share classes or update shared classes an application registers an API by passing an API name, version, and Shared Classes Description File (SCDF) to the implementation. The SCDF is in the same format as the Application Description File specified in Section 12.2.8, and specifies the classes that make-up the registered API. The SCDF specifies all files and directories to be copied. The SCDF SHALL be located in the same transmission file system as the shared files in a base directory that can be used for relative path creation to the shared files. SCDF entries can be converted to absolute paths defined in the same manner as ADF entries; see Section 12.2.8.1. When converted to absolute paths, SCDF entries are relative to the location of the SCDF. For example; if the absolute path of the SCDF is "/apps/RegisteredApi1/sharedfiles.scdf", then the location of the SCDF is "/apps/RegisteredApi1" and all of the SCDF entries are relative to this location when converted to paths. A java.io.File path parameter passed to the org.ocap.system.RegisteredApiManager.register method SHOULD contain the absolute path of the SCDF; otherwise, the implementation may not be able to locate the corresponding shared files. If not all of the required files and directories are specified properly, an application may not be able to use shared classes since authentication will fail. The OCAP implementation shall provide an application with access to the files and directories described in the SCDF in the same storage directory structure as the original structure in the file transmission system. It is allowed that the OCAP implementation store files in a different structure internally as long as it provides the original structure to the application.

> For example, the OCAP implementation may create a new additional top directory (or directories) over the top of copied directory structure to separate from another directory structure. In such case, the OCAP implementation SHALL convert original class path information to the new directory structure. For example, "/com/ocap/Common.class" in an original DSMCC object carousel may be copied to "/copytop/com/ocap/Common.class". In this case, a "/com/ocap" class path in XAIT SHALL be converted to "/copytop/com/ocap". SCDF is the following:

```
<applicationdescription>
    <dir name="com">
        <dir name="ocap">
            <file name="Common.class" size="10"/>
        </dir>
    </dir>
```

```
</applicationdescription>
```

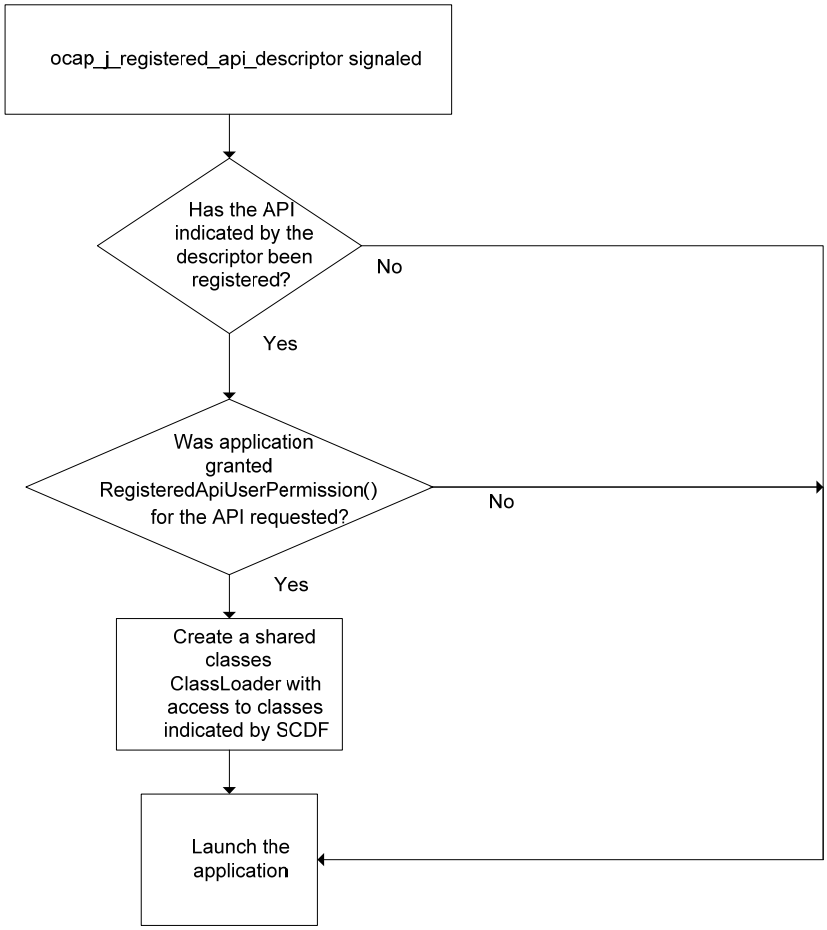
**Note:** A class path may be described in XAIT in a format of "base_dir + relative_classpath_extension" or "absolute_classpath_extension".

To remove a registered API an application passes in the name of the registered API. An application MAY query information about registered APIs as well. However, application access to shared classes can only be modified by application signaling.

#### *21.2.1.21.2Accessing a Registered API*

An application can be signaled with the ocap_j_registered_api_descriptor, in order to access shared classes within a registered API. The ocap_j_registered_api_descriptor contains the name of a registered API the application is requesting access to.

The flow chart below illustrates the actions the implementation SHALL take when an ocap_j_registered_api_descriptor is signaled with an application:




#### *21.2.1.21.3Installation*

When an application calls the org.ocap.system.RegisteredApiManager.register() method, installation of the registered API commences. The installation process SHALL execute the following steps in order:

a. Read SCDF and relate it to the specified API name.

b. Check that shared class and data files are dual-signed and authenticate the shared class and data files using certificate, signature and hash files specified in SCDF. (See also Section 21.2.1.21.8.) In this context, data files are deemed to exclude security related data files: digest (hash) files, signature files, certificate files, for which the special authentication rules specified by Section 14.2.1.15 apply. All class and data files other than these security files SHALL be listed in their governing hashfiles and SHALL specify a digest_code other than zero (non-authenticated).

c. Store files described in SCDF. The application should check if the specified registered API has already been installed before calling the register() method. The getNames() and getUsedNames() methods in the RegisterApiManager provide API names that have already registered.

**Note:** The installation doesn't process class loading. Class loading will be done when an application that uses the registered API is launched.

When installing shared classes, if storage priority specified when registering an API indicates no storage, or persistent storage has insufficient space, the implementation SHALL attempt to install shared classes in an implementation specific location that MAY be volatile; provided such storage is available.

If a class is listed in both the sharing application's application description file and shared classes description file, and the sharing application is stored, the implementation MAY store a single copy of the class. In this case the higher of the storage priorities indicated by the ocap_j_registered_api_descriptor and application_storage_descriptor is used.

A registered API is either completely available, including all classes specified in the SCDF, or it is not available. If a terminal deletes part of a registered API (perhaps because it was in volatile storage and the terminal rebooted), it SHALL un-install and delete the entire registered API.

#### *21.2.1.21.4Access*

For an application to access a registered API, one or more ocap_j_registered_api_descriptors SHALL appear in the application's descriptor loop in the AIT for bound applications or in the XAIT for unbound applications. Any ocap_j_registered_api_descriptors that refer to a registered API that is not installed at the time the application is loaded are ignored for the purposes of this subsection, and such registered APIs SHALL NOT be accessible to the application even if they are installed later. (unless the application is destroyed and re-launched after the registered APIs are installed.)

Shared classes in a registered API SHALL NOT be class loaded until after the application using those classes has been authenticated; (see also Section 21.2.1.21.8). Shared classes in a registered API are accessed via a shared classes class loader. This class loader SHALL be either a new delegation parent of the application class loader, or the application class loader. If it is a new delegation parent it SHALL be created before any of the application's classes are loaded. If it is a new delegation parent it SHALL NOT be created for applications that do not specify any valid ocap_j_registered_api_descriptors, and it SHALL NOT be created if none of the registered APIs listed in the ocap_j_registered_api_descriptors are installed.

The directory that contains an SCDF is the base directory of the shared classes specified by the SCDF. A class path to the base directory of shared classes is automatically added to the class path of each application that uses the shared class. An application can specify a class path in a dvb_j_application_location_descriptor in its own AIT or XAIT application descriptor loop if the directory structure of the shared class is known.

**Note:** If a shared class requires multiple class path elements, an application SHALL specify the class path in its own AIT or XAIT.

Where the shared class loader is the application class loader, the base directory for each registered API SHALL be appended to the application's class path in the order the ocap_j_registered_api_descriptors appear in the descriptor loop.

Where the shared class loader is a new delegation parent of the application class loader, the class path for that new delegation parent SHALL be the base directories for each registered API in the order the ocap_j_registered_api_descriptors appear in the descriptor loop.

The OCAP implementation SHALL search class in the following class path order:

a. base_directory_byte in a copied directory structure according to each referenced SCDF. (See also Section 21.2.1.21.1.)

b. In order of classpath_extension_byte in a copied directory structure according to each referenced SCDF. (See also Section 21.2.1.21.1.)

c. base_directory_byte in a copied directory structure according to the ADF. (See also Section 12.2.8.1.)

d. In order of classpath_extension_byte in a copied directory structure according to the ADF. (See also Section 12.2.8.1.)

e. base_directory_byte in the original file transmission system.

f. In order of classpath_extension_byte in the original file transmission system.

**Note:** The classpath_extension_byte may be concatenated with a base_directory_byte. They are specified in the AIT or XAIT.

The base directory in the host's storage for shared classes is unknown to applications initially. An application needs to use the Class Loader (e.g., the ClassLoader.loadClass(), getResource(), and getResourceAsStream() methods). The ClassLoader.getResource() method SHALL work for resources loaded from a registered API, and the returned resource SHALL be readable in the usual way. The ClassLoader.getResource() method MAY return a URL using the file: protocol, if so then the application SHALL be able to read that file using java.io, but SHALL NOT be able to write to it.

Shared class files cannot be accessed before they have been installed. Applications attempting to instantiate an object instance of a shared class SHALL be thrown a java.lang.ClassNotFoundException if the shared class has not been installed for any reason, and a class with the same name was not included with the application. A shared class file may not have been installed because no application attempted to install it, the class could not be authenticated, or the installing application does not have MonitorAppPermission("registeredapi.manager").

#### *21.2.1.21.5Persistence*

Once shared classes have been installed they can be accessed even after the installing application has been destroyed. Shared classes that have been installed in persistent storage will be accessible after a reboot or power-cycle. Shared classes that have been installed to volatile storage only will not be accessible after a reboot or power-cycle.

#### *21.2.1.21.6Removal*

When a Host device is moved to a different network the implementation SHALL remove registered APIs registered prior to the move. Shared classes SHALL be removed from persistent storage in the same fashion as application files as defined by Section 12.2.6. In addition, any application with MonitorAppPermission("registeredapi.manager") MAY remove a registered API using the org.ocap.system.RegisteredApiManager singleton. Removing a registered API MAY adversely affect applications accessing shared classes within the API.

#### *21.2.1.21.7Update*

An application can update a registered API by trying to register an API with the same name as an application already registered. Unique registered API names SHOULD be used when this is not the intent. When the same name as a registered API is passed in, the implementation SHALL update the registered API if the versions do not match. If the versions do match, the implementation SHALL NOT update the registered API.

#### *21.2.1.21.8Security*

Shared class files SHALL be authenticated with applications that install them as specified in Section 14. The SCDF SHALL contain certificate, signature, and hash files needed to authenticate any shared files.

**Note:** See Section 13.3.1.2 for further information on loading shared class files and shared authenticated files.

For authentication, Section 12.2.7 applies. Shared class files are authenticated once at installation time; subsequent uses of shared class files or other shared authenticated files does not require that further authentication of those files be performed".

Shared classes indicated by an SCDF file SHALL be authenticated with the application that registers the corresponding API. When shared classes are accessed by an application signaled with the ocap_j_registered_api_descriptor, the classes will be granted the same permissions as the application.

Note that there is no package sealing for registered APIs. Applications may define classes in the package(s) used by the registered API, allowing them to access package-private members of registered API classes according to normal Java rules.

### *21.2.1.22 Media Presentation Management*

An application that has MonitorAppPermission("mediaAccess") can implement a MediaAccessHandler to prevent the presentation of A/V service components when a new service is selected or when the conditions of A/V service components presentation changes (for instance, a PMT change).

The registered MediaAccessHandler is invoked each time a MediaPresentationEvaluationTrigger is generated either by the OCAP implementation, or, by an application that has MonitorAppPermission("mediaAccess"), through the MediaAccessConditionControl JMF control. The OCAP implementation SHALL block the new presentation corresponding to the new environment that led to the generation of the trigger until the MediaAccessHandler grants permission. It is implementation dependent, whether presentation of previously selected service components is stopped or not. The OCAP implementation SHALL call the MediaAccessHandler when required for service players bound, or not, to ServiceContext.

**Note:** In order to preserve end user experience, an OCAP application with sufficient privilege should not call the MediaAccessControl.conditionHasChanged() method from the JMF control to ask for revaluation in case one of the conditions to issue a trigger are met, but there is no reason to block the content.

**Note:** Should a monitor application desires not to block presentation upon service selection, the following scenario could be followed. Upon service selection, the monitor application can return immediately from the checkMediaAccessAuthorization() method, checks subsequently whether the new service components can be presented or not and calls the MediaAccessControl.conditionHasChanged() method with a trigger of its own in case the presentation needs to be blocked or changed.

The MediaAccessHandler is registered via a MediaAccessHandlerRegistrar. The monitor application can also indicate to the OCAP implementation which evaluation trigger it will manage exclusively through the MediaAccessHandlerRegistrar.setExternalTriggers() method. In such case the OCAP implementation SHALL NOT generate such type of triggers any more.

Two types of triggers are defined:

- Mandatory triggers: an OCAP implementation SHALL be able to generate such trigger independently of the monitor application. A monitor application cannot manage exclusively such triggers.
- Optional triggers: such triggers MAY be generated by the OCAP implementation. A monitor application can manage exclusively such triggers.

***Table 21–2 - Triggers***

| Triggers leading to a call to the MediaAccessHandler during the revaluation process | Description | Mandatory /Optional |
|---|---|---|
| PMT_CHANGED | The broadcast PMT has changed. OCAP implementation shall not generate such trigger on a new service selection, nor on an update that does not lead to a presentation change (e.g., scrambled with right to free to air service components, not presented service component removed from the broadcast). | Mandatory |
| RESOURCE_AVAILABILITY_CHANGED | Access to a resource has changed: lost or free resource. | Mandatory |
| NEW_SELECTED_SERVICE | A new service has been selected. | Mandatory |
| NEW_SELECTED_SERVICE_COMPONENTS | New service components have been explicitly selected (via JMF control or ServiceContext) or implicitly selected (e.g., following a PMT changed, or a resource availability changed).<br>OCAP implementation shall not generate such trigger on a new service selection. | Mandatory |
| POWER_STATE_CHANGED | The power state has changed, (e.g., switch to power on). | Optional |
| CURRENT_PROGRAM_EVENT_CHANGED | The current program event has changed. The OCAP implementation can generate this trigger in case of SCTE-65 profile 4, 5, and 6. | Optional |
| USER_RATING_CHANGED | User preference for rating has been changed. | Optional |
| PROGRAM_EVENT_RATING_CHANGED | The program event rating has changed. The OCAP implementation can generate this trigger if the content_advisory_descriptor is present in the PMT for the service changes, or if SCTE-65 profile 4, 5, or 6 is used. | Optional |

On trigger generation, the registered MediaAccessHandler is called and shall indicate which service components shall be presented or not. For each service component, the MediaAccessHandler does not authorize presentation; it can provide a list of reasons for the denial.

Thus, the OCAP implementation SHALL notify player listeners of the presented media. A media presentation is defined as normal if the presented service components corresponds to the requested ones. A media presentation is defined as alternative if the presented service components do not correspond to the requested ones.

NormalMediaPresentationEvent SHALL be generated when:

- normal media content presentation begins,
- during the presentation of a service, if an alternative media content was presented and that alternative content is replaced with the normal media content of the service,

- during the presentation of a service, if normal media content was presented and a revaluation leads to the presentation of a new normal media content.

An AlternativeMediaPresentationEvent notification SHALL be generated when:

- alternative media content presentation begins,
- during the presentation of a service, if any of the service components being presented is replaced with an alternative media content,
- during the presentation of a service, if alternative media content was presented and a revaluation leads to the presentation of a new alternative media content.

The AlternativeMediaPresentationEvent implements the NotPresentedMediaInterface in order to report the list of not presented service components and reasons why their presentation failed.

The following table list the possible reasons defined in NotPresentedMediaInterface:

***Table 21–3 - Reasons***

| Reason | Description |
|---|---|
| BROADCAST_INCONSISTENCY | AlternativeMediaPresentationReason indicating that broadcast information is inconsistent: for example PMT is missing. |
| CA_UNKNOWN | AlternativeMediaPresentationReason indicating that media are ciphered and the CA does not correspond to ciphering. |
| COMMERCIAL_DIALOG | AlternativeMediaPresentationReason indicating that a user dialog for payment is necessary before media presentation. |
| HARDWARE_RESOURCE_NOT_AVAILABLE | AlternativeMediaPresentationReason indicating that hardware resource necessary for presenting service components is not available. |
| RATING_PROBLEM | AlternativeMediaPresentationReason indicating that media presentation is not authorized due to invalid rating. |
| NO_ENTITLEMENT | AlternativeMediaPresentationReason indicating that service components are ciphered and the user has no entitlement to view them or part of them. |

For each service Player, an application can specify a MediaTimer and register itself for MediaTimer events. The MediaTimer is defined with a first time and a last time in the media time of the played content. When the Player reaches or goes past or beyond the first time or the last time, the OCAP implementation SHALL notify the registered application.

#### *21.2.1.23 User Preference Setting*

The OCAP implementation shall manage the following preferences as the form of org.dvb.user.GeneralPreference. Applications with MonitorAppPermission ("properties") can change the following preferences.

- Audio Language: 3-letter ISO 639 language codes
- Audio Format: "Dolby" or "MPEG"
- Audio precedence: "Format" or "Language"

Note: The OCAP implementation is responsible for selecting the audio elementary stream based on the three user preference settings listed above. If there are two different audio streams in which one matches only

Language and the other matches only Format, the OCAP implementation should select the default audio based on the "Audio precedence" setting. (If the Audio precedence is set to "Language", the language-matched stream has higher priority than the format-matched stream.)

- Timezone: GMT[+-]hh:mm

Note: Timezone shall be used for the default timezone of the OCAP implementation.

- Analog Audio: "Primary" or "Secondary"

Note: If "Analog Audio" is set to "Secondary", the OCAP implementation is responsible for selecting the Secondary audio stream for the analog service by default.

- Closed Caption On: "On" or "Off"

Note: If "Closed Caption On" is set to "Off", the OCAP implementation shall turn off closed captioning by default.

# 22 OBJECT CAROUSEL

This section is compliant with [DVB-GEM 1.1] Section: Annex B (normative): Broadcast file system and trigger transport and contains extensions of [DVB-MHP1.1.2] Section: Annex B (normative): Object carousel.

This section of the OCAP 1.1 Profile describes the technical differences between the [DVB-MHP1.1.2] object carousel and the OCAP 1.1 object carousel. When the [DVB-MHP1.1.2] is read in the context of OCAP 1.1, all instances of the MHP terminal SHALL be read as the OCAP 1.1 terminal. The numbering of the units of this chapter corresponds to the numbering in [DVB-MHP1.1.2] in order to facilitate the reader's ability to determine changes from that document. The differences documented here are based both on the unique environment created by the cable industry as well as current corrigenda in the DVB-MHP revision process.

Like Annex B of [DVB-MHP1.1.2], this section is based upon the following:

- [ISO 13818-1]—MPEG 2 systems
- [ISO 13818-6]—DSM-CC

Additionally, this section specifies descriptors, some of which can be carried in carousel messages and others of which can be carried in the AIT. Those descriptors are defined and further explained in either [DVB-MHP1.1.2] or in one of the following documents:

- [EN 301 192]—DVB specification for data broadcasting
- [ETSI TR 101 202]—Implementation Guidelines for Data broadcasting

## 22.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

Section 22 (this section) of the OCAP 1.1 Profile corresponds to Annex B of [DVB-MHP1.1.2] as follows:

***Table 22–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 22 Object Carousel | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | Annex B (normative): Object carousel | Extension |
| 22.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 22.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 22.2.1 Deviations from the DVB-MHP Specification | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.1 Introduction | Compliance |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.1.1 Key to notation | Compliance |
| 22.2.1.1 DSM-CC Sections | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.2 Object Carousel Profile | Extension |
| 22.2.1.1 DSM-CC Sections | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.2.1 DSM-CC Sections | Extension |
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.2.1.1 Sections per TS packet | Compliance |
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.2.2 Data Carousel | Compliance |
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.2.2.1 General | Compliance |
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.2.2.2 DownloadInfoIndication | Compliance |
| 22.2.1.2 DownloadServerInitiate | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.2.2.3 DownloadServerInitiate | Extension |
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.2.2.4 ModuleInfo | Compliance |
| 22.2.1.3 ServiceGatewayInfo | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.2.2.5 ServiceGatewayInfo | Extension |
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.2.2.6 Download Cancel | Compliance |
| 22.2.1.4 Object Carousel NSAP | Annex B (normative): Broadcast filesystem and trigger transport | Extension | B.2.3 The Object Carousel | Extension |
| 22.2.1.5 CORBA Strings | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | Annex B.2.3.2, CORBA Strings | Extension |
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.2.4 Broadcast timebases and events | Compliance |
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.2.5 Assignment and use of transactionId values | Compliance |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.2.6 Informative Background | Compliance |
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.2.7 DVB semantics of the transactionId field | Compliance |
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.2.8 Mapping of objects to data carousel modules | Compliance |
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.2.9 Compression of modules | Compliance |
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.2.10 Mounting an Object Carousel | Compliance |
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.2.11 Unavailability of a carousel | Compliance |
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.2.12 Delivery of Carousel within multiple services | Compliance |
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.3 AssociationTag Mapping | Compliance |
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.4 Example of an Object Carousel (informative) | Compliance |
| No Corresponding Section | Annex B (normative): Broadcast filesystem and trigger transport | Compliance | B.5 Caching | Compliance |
| 22.2.2 Extensions to DVB-MHP | No Corresponding Section | OCAP Specific Extension | No Corresponding Section | |

## 22.2 OCAP 1.1 Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

### 22.2.1 Deviations from the DVB-MHP Specification

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

#### *22.2.1.1 DSM-CC Sections*

OCAP 1.1 deviates from [DVB-MHP1.1.2] Section B.2.1, Table B.2 as follows: the value of the last_section_number field MAY have any value in the range 0x00 to 0xFF (inclusive) as defined by [ISO 13818-6], Section 9.2.2.1, and, furthermore, if the value of last_section_number is 0xFF, then an OCAP implementation SHALL NOT abort the retrieval and SHALL NOT report an error condition.

Note: For modules with 256 or more sections, the last_section_number becomes 0xFF as this is the maximum value that is encoded in the section_number field for this module.

#### *22.2.1.2 DownloadServerInitiate*

This subsection is compliant with [DVB-GEM 1.1] Section: Annex B (normative): Broadcast file system and trigger transport and contains extensions of [DVB-MHP1.1.2] Section: B.2.2.3 DownloadServerInitiate.

[DVB-MHP1.1.2] stipulates that the serverID's 20 bytes are set to the value 0xFF. This conflicts with DSM-CC's requirement that this value contain an NSAP address which, for the object carousel, SHALL start with 0x0000. Therefore, OCAP 1.1 modifies this value to the value stipulated in DSM-CC as specified by [ISO 13818-6].

#### *22.2.1.3 ServiceGatewayInfo*

This subsection is compliant with [DVB-GEM 1.1] Section: Annex B (normative): Broadcast file system and trigger transport and contains extensions of [DVB-MHP1.1.2] Section: B.2.2.5 ServiceGatewayInfo.

In addition to the restrictions in [DVB-MHP1.1.2], OCAP 1.1 requires that the first tap in the IOP:IOR SHALL be of type `BIOP_DELIVERY_PARA_USE`.

#### *22.2.1.4 Object Carousel NSAP*

OCAP 1.1 deviates from [DVB-MHP1.1.2] Table B.26 and does not require support for the NSAP format defined therein. Instead implementations SHALL support the NSAP defined in Table 22–2. The scope of an OCAP NSAP address is as follows:

- The ServerID field for a DownloadServerInitiate message SHALL NOT include an OCAP NSAP address, see Section 22.2.1.2 for further information.
- The DSM::ServiceLocation field of the LiteOptionsProfileBody form of an Interoperable Object Reference SHALL contain an OCAP NSAP address in the serviceDomain_data field.
- Any text referenced by OCAP including javadoc and requiring an NSAP parameter or return value SHALL be read as requiring an OCAP NSAP address.

***Table 22–2 - OCAP NSAP***

| Syntax | bits | Type | Value | Comment |
|---|---|---|---|---|
| `OCAPcarouselNSAPaddress() {` | | | | |
| `AFI` | `8` | `uimsbf` | `0x00` | `NSAP for private use` |
| `Type` | `8` | `uimsbf` | `0x00` | `Object carousel NSAP Address` |
| `carousel_id` | `32` | `uimsbf` | `+` | `To resolve this reference a carousel_id_descriptor with the same carousel_id as indicated in this field SHALL be present in the PMT signaling for the service identified below.` |
| `specifierType` | `8` | `uimsbf` | `0x01` | `IEEE OUI` |
| `specifierData {IEEE OUI}` | `24` | `uimsbf` | `0x001000` | `CableLabs Organization` |
| `ocap_service_location () {` | | | | |
| `transport_stream_id` | `16` | `uimsbf` | `0x00` | |
| `original_network_id` | `16` | `uimsbf` | `0x00` | |

| Syntax | bits | Type | Value | Comment |
|---|---|---|---|---|
| service_id | 16 | uimsbf | + | When multiplex type == '01', this field == source_id as per ANSI/SCTE 65 2002 [SCTE 65]. When multiplex type =='10', or multiplex type =='11', this field == program_number as per ISO/IEC 13818-1:2000 [ISO 13818-1]. |
| multiplex_type | 2 | uimsbf | + | '00'==reserved,<br>'01' == inband,<br>'10'==DSG application tunnel,<br>'11' == OOB |
| if (Multiplex_type == '10' { | | | | |
| Reserved | 14 | bslbf | 0x3fff | |
| external application id | 16 | uimsbf | + | Identifies the associated DSG Application ID as per CM-SP-DSG [DSG]. |
| } | | | | |
| else | | | | |
| Reserved | 30 | bslbf | 0x3fffffff | |
| } | | | | |
| } | | | | |
| } | | | | |

#### *22.2.1.5 CORBA Strings*

In addition to the restrictions in [DVB-MHP1.1.2], OCAP 1.1 requires that:

a. All CORBA strings SHALL be interpreted in accordance with the Char Transmission Code Set (TCS-C) as defined by [CORBA/ IIOP] (i.e., no use is made of the wide character string in CORBA string instances).

b. The Char Transmission Code Set used with CORBA strings SHALL be uniform within a single instance of an Object Carousel (i.e., a service domain, and SHOULD be signaled by means of a GIOP Code Set Service Context structure, as defined by [CORBA/ IIOP]), pg. 10-29. If signaled, the GIOP Code Set Service Context SHALL appear exactly once within the service context list loop of the BIOP::ServiceGateway message. When mounting an object carousel instance, an OCAP terminal SHALL determine if a GIOP Code Set Service Context is present in the BIOP::ServiceGateway message, and, if present and valid, SHALL use the signaled Char Transmission Code Set in order to interpret CORBA string instances contained within the message structures of that object carousel instance.

c. If a GIOP Code Set Service Context is signaled, one of the following values SHALL appear in the char_data field of the GIOP Code Set Service Context, with all other values being reserved for future standardization:

- 0x00010001 - ISO 8859-1
- 0x05010001 - UTF-8

d. If a GIOP Code Set Service Context is signaled, the wchar_data field of the GIOP Code Set Service Context SHALL be zero (0).

e. If no GIOP Code Set Service Context is present in the service context list of the BIOP::ServiceGateway message, then the Char Transmission Code Set (TCS-C) SHALL be considered to be ISO 8859-1 for the

purpose of decoding CORBA string instances.

The following (informative) example depicts the signaling of a GIOP Code Set Service Context that specifies that the value of the Char Transmission Code Set (TCS-C) for CORBA strings is UTF-8:

Example

| Field | Bits | Type | Value | Comment |
|---|---|---|---|---|
| | | | | I |
| context_id | 32 | uimsbf | 0x00000001 | OP::ServiceID CodeSets = 1 |
| context_data_length | 16 | uimsbf | 0x0008 | length of CodeSetContext |
| char_data | 32 | uimsbf | 0x05010001 | TCS-C = UTF-8 |
| wchar_data | 32 | uimsbf | 0x00000000 | TCS-W =unknown (i.e., not defined) |

### 22.2.2 Extensions to DVB-MHP

#### *22.2.2.1 OOB/DSG Object Carousel*

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] section.

The [DVB-MHP1.1.2] does not specify handling of an out-of-band object carousel. OCAP 1.1 extends the MHP to define carriage of object carousels out-of-band and received in MPEG flows from the CableCARD. OCAP 1.1 supports a maximum of one OOB object carousel, and consequently one OOB PAT and one OOB PMT. To enable carriage of object carousels in MPEG flows, OCAP 1.1 asserts the following implementation rules:

- When an application attempts to attach to an OOB object carousel, the implementation SHALL attempt to open an OOB MPEG flow on PID 0x0000 and determine if a PAT is present, if it hasn't already done so. If the CableCARD opens the flow and no PAT is detected within two times the maximum PAT repetition rate, the implementation SHALL fail the attach() method as per the org.dvb.dsmcc.ServiceDomain.attach() method specification, and MAY close the flow. An implementation MAY open a MPEG flow for OOB object carousel access before an application attach request.
- The implementation SHALL reserve one Extended Channel MPEG flow for OOB object carousel use. When the flows_remaining field of the new_flow_cnf message indicates 1 flow remaining, and no flow is opened for OOB object carousel, the implementation SHALL NOT allow an application to open another MPEG flow. See [CCIF 2.0], Table 19.4-4 for new_flow_cnf message format.
- In an effort to conserve flows, once a PAT is received the implementation MAY use the same flow for the PMT PID by closing the flow to PID 0x0000 and re-opening the flow to the PMT PID. When one flow is used for both PAT and PMT, if the PMT is not detected within the maximum PMT repetition rate specified in this specification section, the implementation SHALL close the flow for the PMT PID and open a flow on PID 0x0000 and re-establish the PAT.
- When an implementation is receiving a PMT in the OOB and the flow to PID 0x0000 has been closed, if the version number of the PMT changes, a flow to PID 0x0000 SHALL be opened and the PAT re-established for version change determination. The implementation SHALL first close the flow on the PMT PID in this case. When multiple flows are available for PAT and PMT acquisition, the implementation SHOULD maintain one flow on PID 0x0000 and one flow for the PMT PID.
- In an effort to conserve flows the DSM-CC Table sections MAY be placed in the same PID as the OOB PMT. If this is not possible and more than one PID is needed for DSM-CC Table sections, and one or more applications are using other PID[s] in flows needed for object carousel access, the implementation SHALL remove the resources necessary to access the carousel from the applications. In this case the implementation SHALL take MPEG flows from applications in the order of lowest to highest priority application until the resources necessary for carousel access have been met.

- Any PID used for an OOB object carousel supersedes application PID requests and consequently MPEG flows associated with OOB carousel flows SHALL NOT be closed in order to satisfy application requests.

When not signaling elementary audio or video stream types, OOB PAT and PMT maximum Table repetition rates SHALL be 1 second. In all other cases PAT and PMT maximum Table repetition rates are defined by [SCTE 54]. OOB PMT entries based on PID indicate MPEG flows in the Extended Channel.

#### *22.2.2.2 Loss of OOB/DSG Object Carousel*

An OOB object carousel can be lost due to factors such as insufficient MPEG flow resources, or forcible OOB forward data channel tuning caused by an OOB_RX_tune_req() APDU; see the [CCIF 2.0]. In the event an OOB object carousel is lost the implementation SHALL comply with carousel recovery and permanent loss behaviors defined by the [DVB-GEM 1.1] inclusion of [DVB-MHP1.1.2], Section 6.2.5.3. For the purposes of OOB object carousel definition, when regarding [DVB-MHP1.1.2] Section 6.2.5.3 and sections it references, the terms "broadcast" and "AIT" can be substituted with "OOB" and "XAIT" respectively.

#### *22.2.2.3 OOB/DSG Object Carousel Data Access*

When an unbound application is delivered in an OOB object carousel, the base directory specified in the application location descriptor, delivered in the XAIT for the application, SHALL reference the base directory of the application within the carousel. In this case the OOB object carousel SHALL be automatically mounted into the application's file space by the implementation. Calling new DSMCCObject(".") or new java.io.File(".") will instantiate the directory object that refers to the base directory as indicated in application location descriptor. See [DVB-MHP1.1.2], Section 10.9.2 for a definition of the application location descriptor.

An unbound application can access a carousel that it is not carried in by calling the org.dvb.dsmcc.ServiceDomain.attach() method and passing in the OCAP NSAP of the carousel. Since the carousel_id is a unique value within OOB object carousels, the other fields do not need to be determined by an application. When the implementation returns the NSAP for an OOB object carousel based on a org.dvb.dsmcc.ServiceDomain.getNSAPAddress() method call it SHALL ensure all of the fields are filled in correctly.

#### *22.2.2.4 OOB/DSG Object Carousel Locator*

The implementation SHALL create OOB object carousel locators using the BNF "ocap://oobfdc.<program_number>", the "oobfdc" term indicates the forward data channel of the OOB DAVIC or DOCSIS channel. Locators of this form can be returned from the org.dvb.dsmcc.ServiceDomain.getLocator() method after the ServiceDomain is attached to an OOB object carousel.

#### *22.2.2.5 DSG Application Object Carousels*

The [DVB-MHP1.1.2] does not specify handling of object carousels over DSG tunnels. OCAP 1.1 extends DVB-MHP to define carriage of object carousels over DSG application tunnels.

OCAP 1.1 supports multiple object carousels on multiple DSG application tunnels. OCAP 1.1 supports a maximum of one object carousel per application tunnel, and also one PAT and one PMT per application tunnel.

A DSG application tunnel has a DSG Client ID that specifies an Application_ID value. The network can map this DSG Application_ID value to a source_name value in the Source Name Sub-table of the Network Text Table.

An OCAP application can attach to DSG application object carousels using one of two methods. An OCAP URL can be used to specify a source_name value and an OCAP NSAP can be used to directly specify a DSG Application ID.

If the NSAP method is used, then the application is expected to supply a value for the DSG Application ID as well as the MPEG program number and the carousel identifier. For DSG application tunnel carousels the program number is encoded within the service_id field as specified in Table 22–2.

If the OCAP URL method is used then the application is expected to supply only a service_name. In this case the special value of 1 is used as the MPEG program number for the object carousel and the implementation SHALL determine the DSG Application ID using the Network Text Table.

To enable carriage of object carousels in DSG application tunnels, OCAP 1.1 asserts the following implementation rules:

- When an application attempts to attach to a DSG object carousel using a service_name, the implementation SHALL use the service_name to determine a DSG Application_ID value using the Network Text Table (see Table 16–3). If the service_name cannot be found in the table, the implementation SHALL fail the attach() method as per the org.dvb.dsmcc.ServiceDomain.attach() method specification.
- The implementation SHALL use the DSG Application_ID value to open the DSG application tunnel flow. The tunnel packets are is expected to contain only MPEG sections encapsulated with in OCAP DSMCC_DSG_Carousel_Header structures as defined in Annex E of [DSG]. If the tunnel does not contain MPEG sections encapsulated within DSMCC DSG Carousel_Header structures then the implementation SHALL fail the attach() method.
- When the tunnel has been opened and verified to contain MPEG sections the implementation SHALL filter the packets with the PID value of 0 specified in the DSG_ Carousel_Header and determine if a PAT is present. If no PAT is detected in the tunnel within two times the maximum PAT repetition rate, the implementation SHALL fail the attach() method. The implementation SHALL use the PAT to determine the appropriate PMT PID for the given MPEG program number.
- The implementation SHALL filter the packets with the PMT PID value and determine if a PMT is present. If no PMT is detected in the tunnel within two times the maximum PMT repetition rate, the implementation SHALL fail the attach() method. The implementation SHALL use the PMT to determine the appropriate PID value for the object carousel.
- The implementation SHALL expect the MPEG sections carried on the DSG application tunnel with the object carousel PID to be DSM-CC Table sections. The DSM-CC Table sections on the tunnel with the object carousel PID value can be expected to contain exactly one object carousel.
- The implementation SHALL ignore MPEG sections on the DSG tunnel that are not PAT, PMT, or DSM-CC Table sections.

The DSG application tunnel PAT and PMT signaling rate shall be 1 second. DSG application tunnel PMT entries based on PID indicate PID values as signaled within the OCAP DSMCC DSG Carousel_Header defined in Annex E of [DSG].

#### *22.2.2.6 DSG Application Object Carousels Data Access*

An OCAP application can access a DSG application tunnel object carousel by calling the org.dvb.dsmcc.ServiceDomain.attach() method and passing in an OCAP URL locator that specifies a service name that has been mapped by the network to a DSG Application_ID.

An OCAP application can also access a DSG application tunnel object carousel by calling the org.dvb.dsmcc.ServiceDomain.attach() method and passing in an OCAP NSAP that specifies a DSG application ID and an MPEG program number.

In all cases, when the implementation returns the NSAP for an DSG application object carousel based on a org.dvb.dsmcc.ServiceDomain.getNSAPAddress() method call it SHALL ensure all of the fields are filled in correctly.

#### *22.2.2.7 DSG Application Object Carousel Locators*

The implementation SHALL create DSG application object carousel locators using the BNF "ocap://n=<service_name>". The "service_name" term indicates the source_name that has been mapped to a DSG Application_ID by the network. Locators of this form can be returned from the org.dvb.dsmcc.ServiceDomain.getLocator() method after the ServiceDomain is attached to a DSG application object carousel.

The network can map a service name to a DSG Application_ID in the Network Text (see Table 16–3).

# 23 TEXT PRESENTATION

This section contains requirements that are extensions to [DVB-GEM 1.1] Section: Annex D (normative): Text presentation and contains extensions of [DVB-MHP1.1.2] Section: Annex D (normative): Text presentation.

This section addresses the OCAP 1.1 APIs that are used for presenting text and their behavior. It also provides information about how downloaded fonts are associated with applications and accessed by them.

## 23.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section. Section 23 (this section) of the OCAP 1.1 Profile corresponds to Annex D of [DVB-MHP1.1.2] and [DVB-GEM 1.1] as follows:

***Table 23–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 23 Text Presentation | Annex D (normative): Text presentation | Extension | Annex D (normative): Text presentation | Extension |
| 23.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 23.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 23.2.1 Deviations from the DVB-MHP Specification | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| No Corresponding Section | Annex D (normative): Text presentation | Compliance | D.1 Scope | Compliance |
| No Corresponding Section | Annex D (normative): Text presentation | Compliance | D.2 Fonts | A subsection is extended |
| No Corresponding Section | Annex D (normative): Text presentation | Compliance | D.2.1 Embedded fonts | Compliance |
| No Corresponding Section | Annex D (normative): Text presentation | Compliance | D.2.2 Downloaded fonts | A subsection is extended |
| No Corresponding Section | Annex D (normative): Text presentation | Compliance | D.2.2.1 Font technology | Compliance |
| No Corresponding Section | Annex D (normative): Text presentation | Compliance | D.2.2.2 Font index files | A subsection is extended |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 23.2.1.1 Format of File | Annex D (normative): Text presentation | Compliance | D.2.2.2.1 Format of file | Extension |
| 23.2.1.2 Element Semantics | Annex D (normative): Text presentation | Compliance | D.2.2.2.2 Element semantics | Extension |
| 23.2.1.3 Example | Annex D (normative): Text presentation | Compliance | D.2.2.2.3 Example | Extension |
| No Corresponding Section | Annex D (normative): Text presentation | Compliance | D.2.2.3 Name and location of font index files | A subsection is extended |
| No Corresponding Section | Annex D (normative): Text presentation | Compliance | D.2.2.3.1 General | Compliance |
| 23.2.1.4 Name of File | Annex D (normative): Text presentation | Compliance | D.2.2.3.2 Name of file | Extension |
| No Corresponding Section | Annex D (normative): Text presentation | Compliance | D.2.2.3.3 Location | Compliance |
| No Corresponding Section | Annex D (normative): Text presentation | Compliance | D.2.2.4 Specification of fonts at run time | Compliance |
| No Corresponding Section | Annex D (normative): Text presentation | Compliance | D.2.2.4.1 DVB-J | Compliance |
| No Corresponding Section | D.1 Horizontal resoluti52 | Compliance | D.3 Text rendering | Compliance |
| No Corresponding Section | D.2 Text wrapping setting is true | Compliance | D.4 Text mark-up | Compliance |

## 23.2 OCAP 1.1 Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

### 23.2.1 Deviations from the DVB-MHP Specification

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

#### *23.2.1.1 Format of File*

This subsection is compliant with [DVB-GEM 1.1] Section: Annex D (normative): Text presentation and contains extensions of [DVB-MHP1.1.2] Section: D.2.2.2.1 Format of file.

This section extends Section D.2.2.2.1 of the [DVB-GEM 1.1].

The `PublicLiteral` used in the OCAP document type declaration of the XML to specify this DTD SHALL be:

```
"-//OCAP//DTD Font Directory 1.0//EN"
```

The SystemLiteral used in the OCAP document type declaration of the XML SHALL be:

```
"http://www.opencable.com/ocap/dtd/fontdirectory-1-0.dtd"
```

#### *23.2.1.2 Element Semantics*

This subsection is compliant with [DVB-GEM 1.1] Section: Annex D (normative): Text presentation and contains extensions of [DVB-MHP1.1.2] Section: D.2.2.2.2 Element semantics.

This section extends Section D.2.2.2.2 of the [DVB-GEM 1.1].

font: There SHALL be at least one font element per font file included in the font directory.

#### *23.2.1.3 Example*

This subsection is compliant with [DVB-GEM 1.1] Section: Annex D (normative): Text presentation and contains extensions of [DVB-MHP1.1.2] Section: D.2.2.2.3 Example.

This section extends Section D.2.2.2.3 of the [DVB-GEM 1.1].

The OCAP document type declaration SHALL be:

```
<!DOCTYPE fontdirectory PUBLIC "-//OCAP//DTD Font Directory 1.0//EN"
 "http://www.opencable.com/ocap/dtd/fontdirectory-1-0.dtd">
```

#### *23.2.1.4 Name of File*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: Annex D (normative): Text presentation.

This section extends Section D.2.2.3.2 of the [DVB-GEM 1.1].

The name of the OCAP font index file SHALL be:

```
ocap.fontindex
```

# 24 EXTENSIONS

This section contains requirements that are extensions to [DVB-GEM 1.1] Section: Annex H (normative): Extensions.

This section defines the OpenCable Application Platform's policy on extensions to its APIs.

Private protocols and APIs are not precluded by the [DVB-MHP1.1.2]. However they are outside the scope of the [DVB-MHP1.1.2] and are not covered there.

## 24.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section. Section 24 (this section) of the OCAP 1.1 Profile corresponds to Annex H of [DVB-MHP1.1.2] and [DVB-GEM 1.1] as follows:

***Table 24–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| 24 Extensions | Annex H (normative): Extensions | Extension |
| 24.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension |
| 24.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension |

## 24.2 OCAP 1.1 Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

### 24.2.1 Extensions to DVB-GEM (Normative)

Private protocols and APIs MAY extend the technology required by the OCAP 1.1 Profile.

However, extensions SHALL NOT include public or protected members to classes or interfaces in the `org.ocap` name space.

# 25 MINIMUM PLATFORM CAPABILITIES

This section contains requirements that are extensions to [DVB-GEM 1.1] Section: Annex G (normative): Minimum platform capabilities and are extensions of [DVB-MHP1.1.2] Section: Annex G (normative): Minimum Platform Capabilities.

The minimum platform capabilities are those features that SHALL be supported by an OCAP 1.1 terminal. Features which are covered in this section include device capabilities, video presentation capabilities, image processing capabilities, transparency capabilities, and color capabilities. This section also specifies the minimum requirements for audio, video, and font support.

## 25.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section. Section 25 (this section) of the OCAP 1.1 Profile corresponds to Annex G of [DVB-MHP1.1.2] and [DVB-GEM 1.1] as follows:

***Table 25–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| 25 Minimum Platform Capabilities | Annex G (normative): Minimum platform capabilities | Extension | Annex G (normative): Minimum Platform Capabilities | Extension |
| 25.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 25.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 25.2.1 Deviations from the DVB-MHP Specification | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| No Corresponding Section | G.1 Graphics | A subsection is extended | G.1 Graphics | A subsection is extended |
| 25.2.1.1 Device Capabilities | G.1.1 Device resolution for Standard Definition | Extension | G.1.1 Device capabilities | Extension |
| No Corresponding Section | G.1.2 Minimum Colour Lookup Table | Compliance | G.1.2 Video presentation capabilities | Compliance |
| No Corresponding Section | No Corresponding Section | | G.1.3 Image processing capabilities | Compliance |
| No Corresponding Section | No Corresponding Section | | G.1.4 Alpha capabilities | Compliance |
| No Corresponding Section | No Corresponding Section | | G.1.5 Colour capabilities | Compliance |

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| No Corresponding Section | No Corresponding Section | | G.1.6 MPEG I frame and Video drips | Compliance |
| No Corresponding Section | G.2 Audio | Compliance | G.2 Audio | Compliance |
| No Corresponding Section | G.3 Video | Compliance | G.3 Video | Compliance |
| No Corresponding Section | G.4 Resident fonts and text rendering | Compliance | G.4 Resident fonts and text rendering | Compliance |
| 25.2.1.2 Input Events | G.5 Input events | Extension | G.5 Input events | Extension |
| No Corresponding Section | G.6 Memory | Compliance | G.6 Memory | Compliance |
| 25.2.2 Extensions to DVB-GEM (Normative) | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 25.2.2.1 Multi-Window System | G.7 Other resources | Extension | G.7 Other resources | Extension |
| 25.2.2.2 OpenCable Set-top Terminal Core Requirements | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 25.2.2.3 Persistent Storage | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 25.2.3 Extensions to DVB-GEM (Informative) | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 25.2.3.1 General Hardware Configuration | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 25.2.3.2 Peripheral Support | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| 25.2.3.3 Power Control | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |

## 25.2 OCAP 1.1 Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

### 25.2.1 Deviations from the DVB-MHP Specification

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

#### *25.2.1.1 Device Capabilities*

This section extends Annex G.1.1 of the [DVB-MHP1.1.2].

**Note:** Annex G.1.1 of GEM [DVB-GEM 1.1] has a bug that references the wrong bullet point item in MHP Annex G.1.1. A bug report has been sent to the DVB-MUG.

References to the MHP Terminal SHALL be replaced with references to the OCAP 1.1 terminal.

The OCAP implementation is required to support the "Platforms Supporting a Full Multi-Window System" implementation scenario as described in org.havi.ui.HsceneFactory. The implementation scenarios "Platforms

Supporting a Restricted Multi-Window System" or "Platforms Supporting a Single Window System" are not valid choices for OCAP.

**Note:** (Informative): In the description of org.havi.ui.HsceneFactory the term 'Window' is to be interpreted as 'HScene' and is not analogous to the term Window as used in AWT.

The minimum set of required device resolutions that MHP terminals SHALL support is replaced by the following description for OCAP:

- When the video format of the currently presenting service is SD, the OCAP implementation SHALL provide at least the following sets of H*Device logical resolution configurations. The aspect ratio of the visible screen may be subject to hardware restrictions or may change according to the display mode of a TV.

***Table 25–2 - Required Device Resolutions***

| | Set #1 | Set #2 |
|---|---|---|
| HGraphicsDevice | 640x480 | 960x540 |
| HVideoDevice | 720x480<br>(according to ITU-R BT.601) | 720x480<br>(according to ITU-R BT.601) |
| HBackgroundDevice | Either 640x480 or 720x480 | Either 640x480 or 720x480 |

- When the video format of the currently presenting service is HD, the OCAP implementation SHALL provide at least the following sets of H*Device resolution configurations. The aspect ratio of the visible screen may be subject to hardware restrictions or may change according to the display mode of a TV. Display of an I-frame on HBackgroundDevice is not mandatory in this scenario. If the OCAP implementation supports display of an I-frame on HBackgroundDevice in this scenario, 1920x1080 resolution is recommended.

***Table 25–3 - Required Device Resolutions***

| | Set #3 | Set #4 |
|---|---|---|
| HGraphicsDevice | 640x480 | 960x540 |
| HVideoDevice | 1920x1080<br>(according to ITU-R BT.709) | 1920x1080<br>(according to ITU-R BT.709) |
| HBackgroundDevice | No I-frame supported | No I-frame supported |

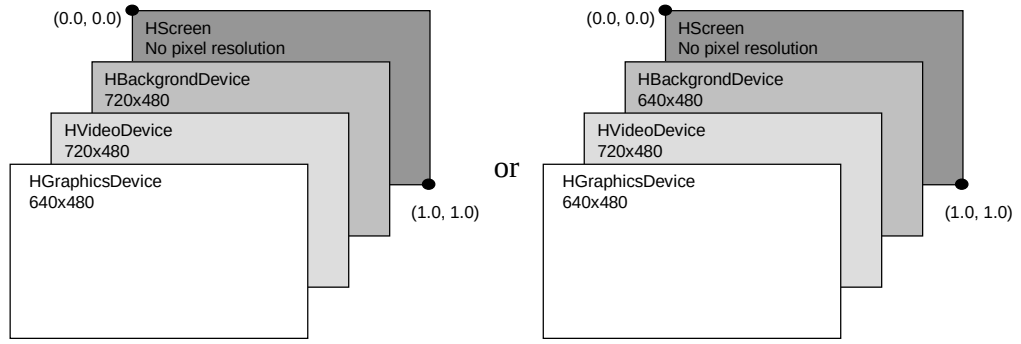
- When the currently presenting service doesn't contain video, the OCAP implementation SHALL provide at least the following sets of H*Device resolution configurations. The aspect ratio of the visible screen may be subject to hardware restrictions or may change according to the display mode of a TV. If the input resolution of the video being processed by the HVideoDevice changes or configuration of the HVideoDevice changes, either due to an application initiated or a platform initiated action, then (1) the configuration of HBackgroundDevice MAY change and (2) if an image is presently displayed in the HBackgroundDevice, then the image MAY no longer be displayed. Note that HVideoDevice configuration could be any resolution and may display solid blue or transparent video plane instead of video contents. When no video is being presented, an application may apply a video transformation to ensure that a background I-frame is visible.

***Table 25–4 - Required Device Resolutions***

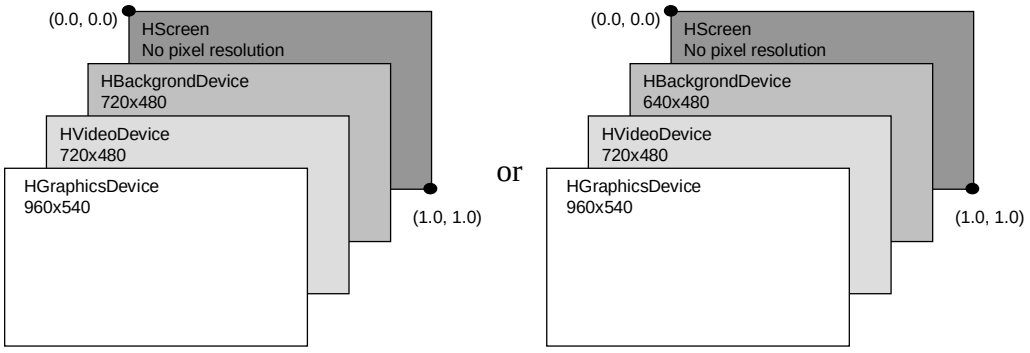
| | Set #5 | Set #6 | Set #7 | Set #8 |
|---|---|---|---|---|
| HGraphicsDevice | 640x480 | 640x480 | 960x540 | 960x540 |

| | Set #5 | Set #6 | Set #7 | Set #8 |
|---|---|---|---|---|
| HVideoDevice | No video input | No video input | No video input | No video input |
| HBackgroundDevice | Either 640x480 or 720x480 | 1920x1080 | Either 640x480 or 720x480 | 1920x1080 |

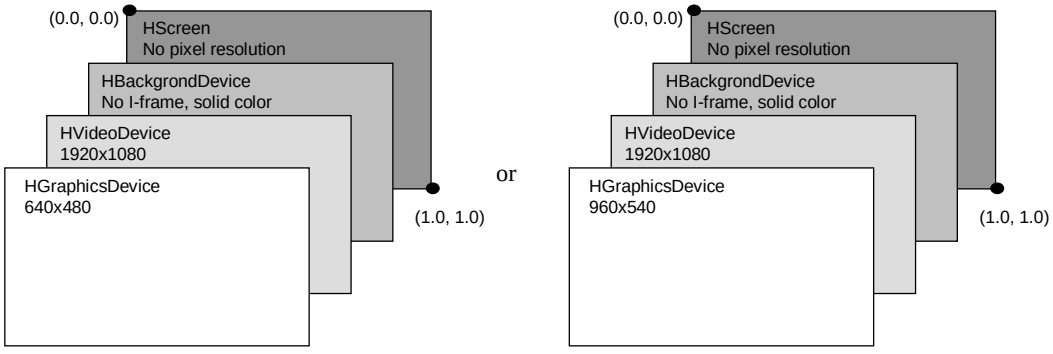
Figure 25–1 illustrates these sets of configurations:




Set #1: 640x480 HGraphicsDevice when SD video input




Set #2: 960x540 HGraphicsDevice when SD video input




Set #3: 640x480 HGraphicsDevice for HD video input

Set #4: 960x540 HGraphicsDevice for HD video input

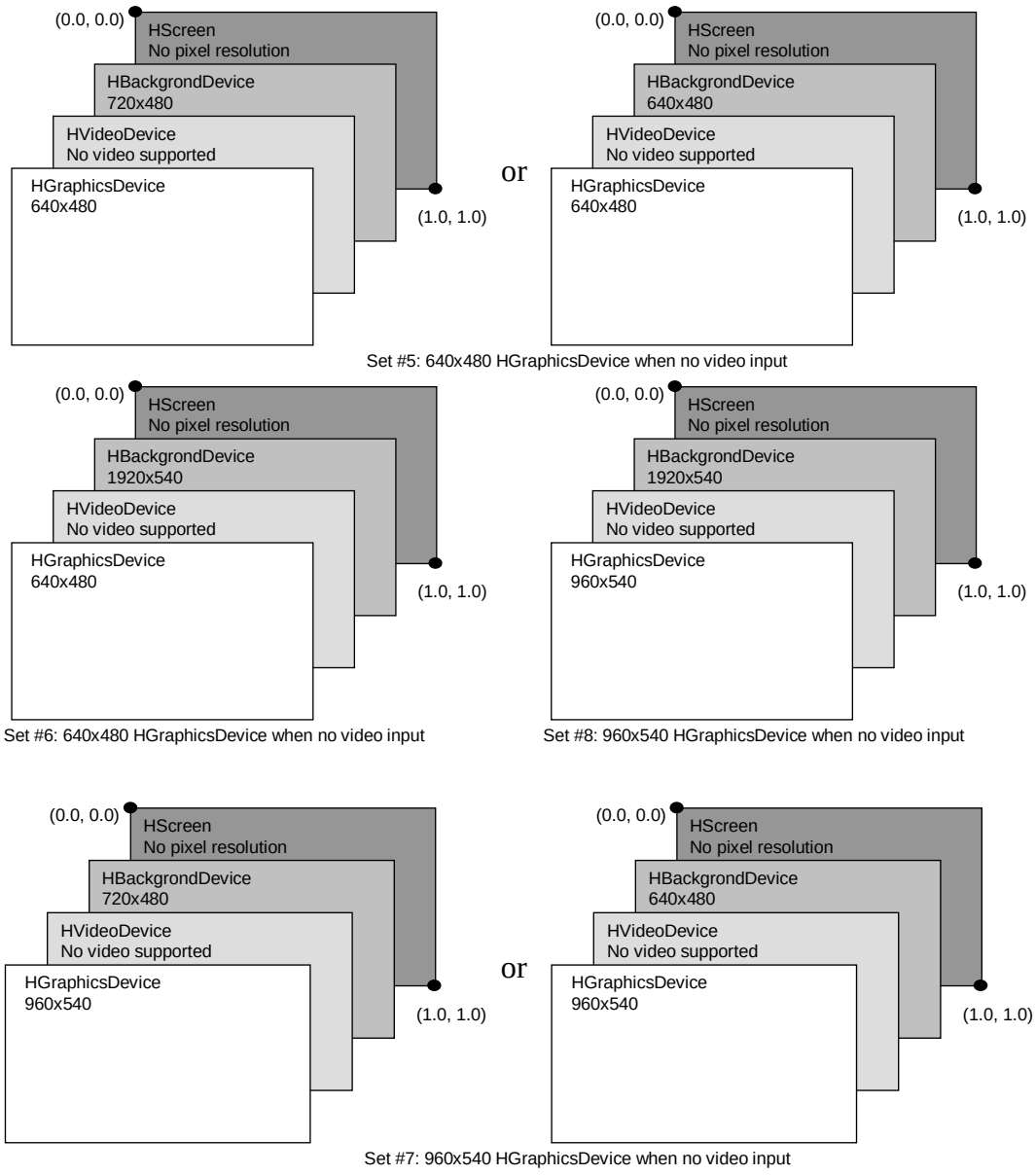



***Figure 25–1 - Required Device Resolutions***

An OCAP implementation SHALL display an I-frame across the full extent of the display rectangle when the resolution of the I-frame matches the resolution of the `HBackgroundDevice` and the `HBackgroundDevice` is configured to be full screen. An I-frame on an `HBackgroundDevice` SHALL be able to be simultaneously displayed with video on an `HVideoDevice`, while decoding of an I-frame while simultaneously decoding video is an implementation option.

**Note:** An application may get the current resolution of `HBackgroundDevice` via `HbackgroundConfiguration` in order to enable display of an I-frame full screen.

It is allowed to provide other sets of configurations in addition to the sets of the minimum configurations above.

If an `HGraphicsDevice` is currently reserved by an application, then the OCAP implementation SHALL NOT automatically change the configuration of the `HGraphicsDevice`, even in the case that input video or the configuration of the HVideoDevice changes.

**Note:** Once an application reserves an `HGraphicsDevice` object, the application can select a graphics configuration of preferred resolution via a `setGraphicsConfiguration()` method. The `HGraphicsDevice` is managed by OCAP resource management described in Section 19. An `HGraphicsConfiguration.getPixelResolution()` method returns graphics resolution of the configuration.

#### *25.2.1.2 Input Events*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: G.5 Input events and are extensions of [DVB-MHP1.1.2] Section: G.5 Input events.

This section extends Annex G.5 of [DVB-GEM 1.1]:

- OCAP 1.1 extends the full set of HAVi remote key events. These extensions are defined in Annex E. The minimal set of key events codes that SHALL be supported for OCAP 1.1 is defined in Table 25–5. (i.e., Table 25–5 replaces Table G.3 of [DVB-MHP1.1.2].)
- All occurrences of "MHP" are replaced with "OCAP". All occurrences of "DVB-J" are replaced with "OCAP-J".
- Note 2 is modified. Only the first sentence is compliant with OCAP. Events generated by the platform that are not listed below are to be sent to the application with focus.
- See Section 16.2.2.1 for details.
- System keycodes represent functionality of the Host device. The Host device is expected to take action on receipt of a system event and to pass the event to the application with focus.
- Table 25–5 represents the set of keycodes that SHALL be present on an input device and sent to OCAP applications.

***Table 25–5 - Minimum Keycode Set***

| Key | KeyEvent | Mandatory Ordinary Keycodes | System Keycodes |
|---|---|---|---|
| Power On/Off | `VK_POWER` | | X |
| Channel Down | `VK_CHANNEL_DOWN` | | |
| Channel Up | `VK_CHANNEL_UP` | | |
| 0-9 | `VK_0` to `VK_9` | X | |
| Up | `VK_UP` | X | |
| Down | `VK_DOWN` | X | |
| Left | `VK_LEFT` | X | |
| Right | `VK_RIGHT` | X | |
| Select | `VK_ENTER` | X | |
| Volume Down | `VK_VOLUME_DOWN` | | X |
| Volume Up | `VK_VOLUME_UP` | | X |
| Mute Volume | `VK_MUTE` | | X |
| Pause | `VK_PAUSE` | | |
| Play | `VK_PLAY` | | |
| Stop | `VK_STOP` | | |
| Record | `VK_RECORD` | | |
| Fast Forward | `VK_FAST_FWD` | | |

| Key | KeyEvent | Mandatory Ordinary Keycodes | System Keycodes |
|---|---|---|---|
| Rewind | VK_REWIND | | |
| Electronic Program Guide | VK_GUIDE | | |
| RF Bypass (Only if Host device has Channel 3/4 RF output) | VK_RF_BYPASS | | X |
| Menu | VK_MENU | | |
| Info | VK_INFO | | |
| Exit | VK_EXIT | | |
| Last | VK_LAST | | |
| Function Key 0 | VK_COLORED_KEY_0 | | |
| Function Key 1 | VK_COLORED_KEY_1 | | |
| Function Key 2 | VK_COLORED_KEY_2 | | |
| Function Key 3 | VK_COLORED_KEY_3 | | |
| Page Up | VK_PAGE_UP | X | |
| Page Down | VK_PAGE_DOWN | X | |
| Next Favorite Channel | VK_NEXT_FAVORITE_CHANNEL | | |
| On Demand | VK_ON_DEMAND | | |

[HOST 2.0] contains requirements regarding support for keyboard input devices, when present. In the presence of a keyboard, a minimum set of keyboard events SHALL be made available to OCAP applications. Table 25–6 represents the set of keycodes that SHALL be present on a keyboard input device, when available, and sent to OCAP applications. Keyboard events not included in Table 25–6 MAY be sent to OCAP applications.

***Table 25–6 - Minimum Keyboard Keycode Set***

| Key | KeyEvent |
|---|---|
| Enter | VK_ENTER |
| Back Space | VK_BACK_SPACE |
| Tab | VK_TAB |
| Shift | VK_SHIFT |
| Control | VK_CONTROL |
| Alt | VK_ALT |
| Caps Lock | VK_CAPS_LOCK |
| Escape | VK_ESCAPE |
| Space | VK_SPACE |
| Page Up | VK_PAGE_UP |
| Page Down | VK_PAGE_DOWN |
| End | VK_END |
| Home | VK_HOME |
| Left | VK_LEFT |

| Key | KeyEvent |
|---|---|
| Up | VK_UP |
| RIGHT | VK_RIGHT |
| DOWN | VK_DOWN |
| , | VK_COMMA |
| . | VK_PERIOD |
| / | VK_SLASH |
| 0 - 9 | VK_0 - VK_9 |
| ; | VK_SEMICOLON |
| = | VK_EQUALS |
| A - Z | VK_A - VK_Z |
| [ | VK_OPEN_BRACKET |
| \ | VK_BACK_SLASH |
| ] | VK_CLOSE_BRACKET |
| 0 - 9 | VK_NUMPAD0 - VK_NUMPAD9 |
| F1 | VK_F1 |
| F2 | VK_F2 |
| F3 | VK_F3 |
| F4 | VK_F4 |
| DELETE | VK_DELETE |
| HELP | VK_HELP |
| QUOTE | VK_QUOTE |

### 25.2.2 Extensions to DVB-GEM (Normative)

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

The following requirements are normative for the OCAP 1.1 Profile. They are specific to OCAP 1.1 and are not contained within the [DVB-MHP1.1.2].

#### *25.2.2.1 Multi-Window System*

This subsection contains requirements that are extensions to [DVB-GEM 1.1] Section: G.7 Other resources and are extensions of [DVB-MHP1.1.2] Section: G.7 Other resources.

The OCAP implementation is required to support the "Platforms Supporting a Full Multi-Window System" implementation as described in Section 25.2.1.1.

##### *25.2.2.1.1 Focus Handling Rules*

Applications may gain input focus as described in Section 13.3.5.2. The implementation SHALL withdraw focus from an application when any of the following events occur:

- The application is moved from the Active state to any other state;
- The application's HScene is hidden, or disposed of.

When the implementation withdraws input focus from an application, the implementation SHALL give input focus according to the following rules:

- The implementation shall maintain an ordered list of activable HScene. An activable HScene is a visible and active (as described in HScene.setActive javadoc) Hscene which has explicitly requested focus at least one time. When an application transitions out of the Active state its HScene SHALL be removed from the activable HScene list.
- The focused HScene is an activable HScene that got the input focus. There is only one focused HScene at a time.
- When a HScene becomes activable, it is added at the end of the list of activable HScene.
- When an activable HScene get focused, it is moved at the beginning of the activable HScene list.
- When a HScene stops being activable, it is removed from the activable HScene list.
- When the focused HScene stops to be activable, the implementation shall withdraw focus from the focused HScene and give input focus to the first activable HScene in the activable HScene list.
- When the first activable HScene is inserted in the activable HScene list, the implementation shall not automatically give input focus to it.

#### *25.2.2.2 OpenCable Set-top Terminal Core Requirements*

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

The resources as specified for all OCAP profiles in [HOST 2.0] are REQUIRED.

#### *25.2.2.3 Persistent Storage*

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

The OCAP implementation SHALL provide a mechanism for logically separate storage of:

- the platform's image as defined in [CCIF 2.0], Section 9.19.
- the root certificates
- the user preferences
- the stored versions of applications, and
- the file store accessible from the OCAP persistent file API in persistent memory.

Persistent memory is any storage medium which retains its contents unaltered between power cycles. Some examples of persistent memory are Flash memory, hard drives, EEROMs, battery-backed RAM.

This feature is required in order to support implementation download, (see Section 20.2.4.7), persistent storage APIs, (see [DVB-MHP1.1.2], Section 11.5.6), and boot-up processing. (See Section 20.2.2, step #6.)

### 25.2.3 Extensions to DVB-GEM (Informative)

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

The following information is informative to the OCAP 1.1 Profile. It attempts to clarify issues in the [DVB-MHP1.1.2], as well as provide additional information to the normative requirements above.

The OCAP 1.1 terminal resources identified in this section are OPTIONAL. These resources are in excess of those resources required to maintain the core functions defined in [HOST 2.0].

#### *25.2.3.1 General Hardware Configuration*

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

OpenCable recommends the following hardware configuration with respect to processor and memory capabilities.

##### *25.2.3.1.1 Processor Capability*

An OCAP 1.1-compliant terminal will most likely require a processor capable of executing instructions at speeds of 130 MIPS or faster. This resource is in excess of those cycles needed to run any manufacturer delivered software including OS and OCAP 1.1, and also in excess of processor usage in the provision of core functionality described in [HOST 2.0]. This capability includes support for video demux, decode and presentation.

##### *25.2.3.1.2 Minimum Memory Configuration*

An OCAP 1.1-compliant terminal will need to provide some amount of memory in excess of any configuration required for OCAP 1.1, including video decoding and graphics display.

25.2.3.1.2.1 RAM

The implementation SHALL provide a minimum of 64 MBs of memory to the cable environment. This memory SHALL be available regardless of the state of the environment.

In order to be able to execute OCAP conformance tests, the following minimum memory requirements are defined for OCAP terminals. All of these are to be measured during normal usage and operational conditions of an OCAP terminal.

All are to be measured in the initXlet() method of a OCAP-J application which is both the only auto-start application signaled in a service and the only application running at that time. The implementation SHALL:

> Successfully load any arbitrary 1,048,576 bytes of Java class files into the memory space of the Java virtual machine. Execution of code called as part of initializing fields in classes is excluded from consideration as part of "load"ing here. RAM usage by the bytecode verifier is included in consideration as part of "load"ing here.

The implementation SHALL supply enough memory to do the above and individually each of the following:

a. Successfully create a Java byte array of lengths from 1 entry to 60,817,408 (58M) entries.

b. Successfully load & display any 46 640x480 32-bit PNG images (conforming to DVB-MHP 1.0.3 [9]DVB-MHP 1.0.3 [9] Section 15.1, "PNG - restrictions" on page 230) from a file that contains only the mandatory information and excludes any optional extension fields or chunks.

c. Successfully load from a file into memory 3456 seconds of audio encoded at 128 kbit/s (where kbit/s is as used in ETSI TR 101 154 [9]). It shall be measured using files that do not include any optional extension fields.

d. Successfully allocate an array of java.lang.Object objects of length 3,538,944 and fill each element of this array with a distinct instance of java.lang.Object.

The memory requirements detailed in this section are not exhaustive. For example, the specific requirement concerning an array of type byte in no way implies that OCAP terminals are exempt from requirements found elsewhere in the OCAP specification (including normatively referenced specifications) for supporting arrays of other types.

OpenCable recommends the use of Zero or One Wait State DRAM or SDRAM, or better.

25.2.3.1.2.2 ROM

Flash Memory SHOULD be partitioned in 128 KByte sectors. Optional protection of Boot Sectors is acceptable to reduce implementation cost. There SHALL be 32 MBs of persistent storage for application code and data (Flash, disk, etc.) available.

25.2.3.1.2.3 NVRAM

OpenCable recommends that the OCAP 1.1 terminal provides for NVRAM with single byte modify/erase capability. The NVRAM SHOULD provide a minimum of 2 Kbytes storage.

### *25.2.3.2 Peripheral Support*

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

OpenCable recommends support for the following peripheral devices.

#### *25.2.3.2.1 Keyboard Device Capabilities*

OpenCable recommends support for keyboard input. If supported, the keyboard input events as identified in the `java.awt.event.KeyEvent` class, as specified in [PBP1.1], SHALL be supported.

#### *25.2.3.2.2 IR Remote Support*

OpenCable recommends support for infrared remote control devices. Both unidirectional and bi-directional devices SHOULD be supported.

### *25.2.3.3 Power Control*

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.1] Section.

#### *25.2.3.3.1 Power Input*

OpenCable requires support for query of the AC power supplied to the OCAP 1.1 terminal. An event SHALL be provided whenever a transition from one state to another occurs. This feature reflects the need for some applications to be able to determine if the terminal is in **standby** mode or is on. When the terminal is on, its full facilities are available. When the terminal is in **standby** mode, the audio, video, and HDNI outputs are inactive. OCAP applications SHALL NOT be terminated while the terminal is in standby mode, for purposes such as EPG update, software maintenance, and quick refresh, when normal power mode is re-established.

# 26 METRICS

This section defines requirements for metrics collection. There is no corresponding section in GEM or MHP.

Metrics are a set of well defined data that are reported to a cable network by a receiver, for the purpose of measuring certain aspects of cable service delivery. OCAP includes specification of a data model and transmission protocol for delivery of metrics to a cable network, and semantic definition of data that is syntactically defined by the data model. The metrics transmission protocol is IPDR/SP.

OCAP implementations SHALL implement the data model and transmission protocol, and all other requirements defined by [Metrics].

## 26.1 Metrics API

OCAP implementations SHALL support the metrics API. The metrics API definition is not included in this version of OCAP, but is expected to by added in a revision by Q1 2007. The API is expected to be JSR 190.

## 26.2 Definitions of metrics data model

This section defines OCAP specific semantics for metrics data elements and conditions under which metrics data SHALL be generated.

### 26.2.1 Conditions which generate metrics

This section describes the specific conditions under which the following metrics records SHALL be generated.

The createTime of a record SHALL equal the time at which the event or operation occurred which resulted in the generation of the record.

The platform MAY be configured to enable and disable any combination of these following record types. When enabled, the following requirements are described for record generation.

| | |
|---|---|
| **serviceSelection** | This record SHALL be generated whenever an OCAP service is selected, either directly by an application or on behalf of an application. |
| **tuning** | This record SHALL be generated whenever a tuner is engaged to acquire a new transport. |
| **SDV** | This record SHALL be generated whenever a Switched Digital Service is requested. |
| **servicePresentation** | This record SHALL be generated whenever video content begins being presented, and when any presentation configuration settings change. |
| **segmentPresentation** | This record SHALL be generated whenever a segmentation descriptor is received that is the first is a series of descriptors for a segment. OUT OF SCOPE until OCAP supports segmentation descriptors. |
| **DPI** | This record SHALL be generated whenever a DPI switch operation occurs. |
| **inputEvent** | This record SHALL be generated whenever an input event is received. |

| | |
|---|---|
| **applicationLoadRequested** | This record SHALL be generated whenever a request to load an application occurs. |
| **applicationLoaded** | This record SHALL be generated whenever an application is loaded. |
| **applicationLaunched** | This record SHALL be generated whenever an application is launched. |
| **applicationPaused** | This record SHALL be generated whenever an application is paused. |
| **applicationResumed** | This record SHALL be generated whenever a paused application resumes running. |
| **applicationTerminated** | This record SHALL be generated whenever an application is terminated. |
| **environmentSelection** | This record SHALL be generated whenever an application environment is selected. This will only occur on an OCAP device that supports multiple application environments. |
| **application** | This record SHALL be generated whenever on application via an API call. |

### 26.2.2 Semantic definition of metrics data model

Wherever the following elements are defined by the data model [Metrics], the following semantics SHALL be used.

| | |
|---|---|
| **protocolVersionModel** | This field SHALL be set 1. |
| **protocolVersionModel** | This field SHALL be set 0. |
| **hostID** | This field SHALL be set by a privileged OCAP application. |
| **hostType** | This field SHALL have the value 'OCAP'. |
| **sourceID** | This field SHALL contain the source_id of the selected service. |
| **tunerID** | This field SHALL be TBD. |
| **aspectRatio** | This field SHALL be encoded as follows; a value of 0 indicates that the service is presenting Standard Definition 4:3, a value of 0 indicates that the service is presenting High Definition 16:9. |
| **resolution** | This field SHALL be TBD. |
| **audio** | This field SHALL have a value of 0 if audio output is not accompanying presentation of associated video, and 1 if audio is output. |
| **segmentID** | This field SHALL TBD. OCAP currently does not support DVB-SAD or SCTE 35 segment descriptors. |
| **replacementID** | This field SHALL have the value of the program number of the MPEG-2 program containing components that are displayed as the result of a DPI switch.. |
| **replacementSource** | This field SHALL be encoded as follows: if a switch has occurred to program components that are broadcast on a different MPEG-2 transport then the original program, the value SHALL be 'Out-of-Transport', if both the original and |

| | |
|---|---|
| | inserted programs are within the same transport, the value SHALL be 'In-Transport'. |
| **switchLevel** | This field SHALL have the value of 'L0' is the switch is an L0 switch, and 'L1' if the switch is an L1 switch. |
| **inputEvent** | This field SHALL have the hexadecimal value of the OCAP KeyCode which generated the event. |
| **appID** | This field SHALL be encoded as defined in OCAP for (X)AIT signaling. |
| **numClassFiles** | This field SHALL be set to the number of class files that have been loaded for an application. |
| **transportProtocol** | This field SHALL be encoded as follows: if the application has been loaded via in band carousel the value SHALL be 'IBOC'; loaded via out of band carousel the value SHALL be 'OOBOC'; if via HTTP the value SHALL be 'HTTP'. |
| **controlCode** | This field SHALL be encoded as follows: if the initial state of the application is LOADED the value SHALL be 'LOADED', if PAUSED the value SHALL be 'PAUSED', if RUNNING the value SHALL be 'RUNNING'. *TBD: check states in OCAP.* |
| **environment** | This field SHALL be encoded as follows: if the selected environment is cable the value SHALL be 0, if non-cable the value SHALL be 1. |

# 27 DIGITAL PROGRAM INSERTION (DPI)

This section defines requirements for Digital Program Insertion, henceforth referred to as DPI. There is no corresponding section in GEM or MHP.

## 27.1 DPI Overview

The digital program insertion (DPI) standards within SCTE (SCTE 35 2004 and SCTE 30 2001) have enabled cable operators to provide local ad insertion on digital content at a head-end level, using equipment available from multiple vendors. OCAP DPI enables similar functionality for DPI on OpenCable Hosts. This enables addressable commercials, or other content, to be inserted at a device level.

To allow for programmability by an MSO as well as meet time sensitive insertion requirements, DPI functionality is separated into two logical components: a 'switch engine' implemented as part of the OCAP platform that performs real-time functions of switching between programs and program elements, and one or more 'targeting engine' OCAP applications that control the behavior of the switch engine.

A switch engine performs switch operations to present inserted content. In the broadcast scenario, which is the only scenario defined herein, inserted content MAY be within the same MPEG-2 Transport Stream as the original program or broadcast within a different MPEG-2 Transport Stream. Other scenarios, such as those where the original or inserted content are available on a storage device, may be defined elsewhere, such as within the OCAP DVR extension.

A switch engine implements two distinct types of switches; a Level 0 (L0) switch, which appears to the viewer as a channel change, and a Level 1 (L1) switch, which is totally invisible to the viewer. During an L0 switch, a switch engine may be required to tune to a different channel to access an insertion stream. An L1 switch never requires tuning; in this case, all of the insertion streams are within the same Transport Stream as the original program content.

Requirements on content to enable L0 and L1 switches are defined by [DVS-766].

Detailed requirements on OCAP to perform L0 and L1 switches are defined in Annex X.

An illustration of the logical components of the OCAP DPI capability is provided in Annex X.

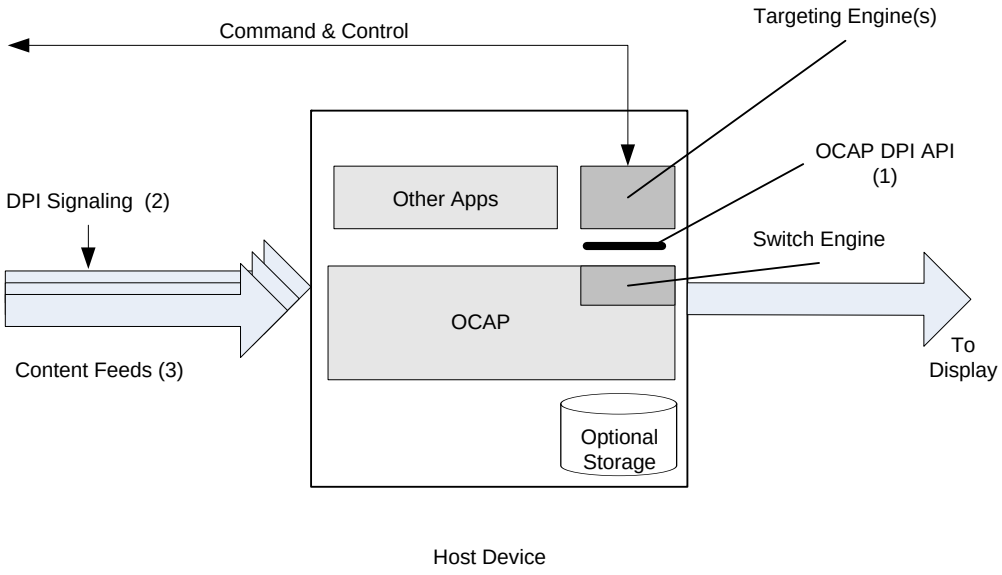



***Figure 27–1 - OCAP DPI Overview***

Indicated as (1) in Figure 27–1, a targeting engine API is defined Annex Xto enable applications to provide the switch engine with instructions and to receive status information on previously submitted instructions.

Indicated as (2) in Figure 27–1, DPI signaling are messages that are processed by the switch engine. Signaling may be in the form of DPI triggers or timelines. See Section 27.2 below for detailed requirements on processing DPI signaling.

Where instructed by a targeting engine, the switch engine performs a program insertion operation on receipt of a DPI trigger, or based on a timeline.

Indicated as (3) in Figure 27–1, content feeds are MPEG-2 Transport Streams that carry program feeds and may contain other insertion feeds, such as advertising feeds. The program feeds (as well as the separate insertion feeds) may contain DPI signaling that is used by the switch engine to insert content, such as addressable commercials during addressable breaks. The program feeds and insertion feeds may be subject to certain encoding (and other) restrictions that make it easier for the switching engine to switch to and from insertion content, see [DVS-766].

The command and control element shown in Figure X represents a hypothetical private communication channel between a targeting engine and an associated network component. Advanced advertising systems are expected to incorporate logical components within the cable network that manage both the insertion of DPI signaling and the actions of a targeting engine.

## 27.2 DPI Signaling

This section describes signaling used to enable a switch engine to perform its operations.

Two signaling protocols are used to support DPI; DPI triggers and DPI timelines. DPI trigger messages embedded within a content segment indicate opportunities to perform a switch operation. DPI timelines are messages embedded within a content segment to enable the implementation to maintain a metadata timeline for a segment. Switch operations MAY be based on timelines.

All DPI signaling is expected to be carried on a single data PID within the original content stream. This data stream SHALL be identified by DPI registration descriptor in a program's PMT.

### 27.2.1 DPI Signaling Descriptor

The `dpi_signaling_descriptor` is defined for use in the elementary stream information loop of the PMT. This descriptor indicates that the associated elementary stream contains a DPI signaling stream, containing DPI triggers and timeline messages as defined below. Only one elementary stream signaled by the PMT SHALL contain a `dpi_signaling_descriptor`. In the event that more than one PMT entry contains a `dpi_signaling_descriptor`, the implementation SHALL ignore all but the first instance listed in the PMT's elementary stream information loop.

A PMT entry with a `dpi_signaling_descriptor` MAY be associated with astream_type of 0xC0 or 0x05.

The `dpi_signaling_descriptor` is defined in Table 27–1.

***Table 27–1 - DPI Signaling Descriptor Syntax***

| Syntax | Bits | Mnemonic |
|---|---|---|
| `dpi_signaling_descriptor() {` | | |
| `  descriptor_tag` | 8 | uimsbf |
| `  descriptor_length` | 8 | uimsbf |
| `  For (i=0; i<n; i++) {` | | |
| `  private_use` | 8 | bslbf |
| `  }` | | |
| `}` | | |

**descriptor_tag** This 8-bit integer with value `0xA2` identifies this descriptor.

**descriptor_length** This 8-bit integer indicates the number of bytes following the descriptor length field.

**private_use** This field MAY be used to carry private data to a receiver which interprets this signaling stream. Its use is not defined by this specification.

### 27.2.2 DPI Triggers

The format for DPI triggers SHALL be the DVB-SAD synchronized event descriptor, see [DVB-SAD] section 5.2.5. When present in a content or insertion feed, the implementation SHALL process every instance of a DVB-SAD synchronized event descriptor.

The use of DPI triggers is different for L0 and L1 switches. To signal an L0 switch, a single DPI trigger MAY be used to indicate the point in a stream at which the switch may occur. For L1 switches, a DPI trigger SHALL be present for each component that will be switched; typically, one for video and one for audio. For L1 switches, a set of DPI triggers SHALL be used to signal a single switch operation.

The following semantics are defined for the DVB-SAD synchronized event descriptor:

**synchronized_event_context:** This field SHALL be encoded according to Table 27–2. If this field contains a value not defined below, the implementation SHALL ignore this descriptor.

*Table 27–2 - synchronized_event_context*

| Value | Description |
|---|---|
| 10 | L1 video |
| 11 | L1 audio |
| 12 | L0 |

**synchronized_event_id** This field identifies a switch operation. The implementation SHALL use this value to associate a switch instruction with this trigger, if such an instruction has been received from an application.

**synchronized_event_data_byte** This field SHALL be encoded according to table Y. If this field is not conformant to Table 27–3, the implementation SHALL ignore this descriptor.

*Table 27–3 - synchronized_event_data_byte*

| Syntax | No. of bits | Identifier |
|---|---|---|
| set_size | 8 | uimsbf |
| position_in _set | 8 | uimsbf |

**set_size** This field indicates the number of triggers signaling a switch. For an L0 switch, this value is 1, for an L1 switch, this value is at least 1 and is typically 2; one for video, one for audio.

**position_in_set** This field indicates the position of this trigger within a set of triggers, zero based.

DPI triggers for L0 switches SHALL be present in a stream at a point TBD.

DPI triggers for L1 switches SHALL be present in a stream at a point TBD. For more details on the structure of an L1 conditioned stream, see [DVS-766]

### 27.2.3 DPI Timelines

A content segment MAY contain timeline messages that allow the implementation to maintain a timeline synchronized to the video and audio content. Timeline messages SHALL be in the DVB-SAD broadcast timeline descriptor format, see [DVB-SAD] section 5.2.2 for a description of the broadcast timeline descriptor. When present within a content feed, the implementation SHALL process every instance of a DVB-SAD broadcast timeline descriptor to maintain an interpolated metadata timeline. [DVB-SAD] contains descriptions and references to definitions of timelines within several sections. The reader should be familiar with [DVB-SAD] in its entirety.

L0 switch operations MAY be performed when a point on a metadata timeline is reached.

In order to enable time based switch scenarios in which a viewer may tune away then tune back into a service during a switch opportunity, streams MAY contain a series of DPI timeline messages throughout the original program.

Within a content feed that signals a timeline and supports L0 switches, a DPI timeline SHALL be continuous throughout the relevant segment. The associated insertion feeds SHALL signal a timeline that is proceeding at the same rate with the same values as the original feed.

Within a content feed that signals a timeline and supporting L1 switches, the DPI signaling stream SHALL contain a continuously advancing timeline.

The following semantics are defined for the DVB-SAD synchronized event descriptor:

Where broadcast_timeline_type equals 0:

**tick_format** This field SHALL have the value 0x11. See [DVB-SAD] table 6.

## 27.3 Switch Engine Behavior

Switches may be trigger based or time based. A switch instruction MAY identify a set of triggers, and or identify a segment of time during which a switch may occur. A switch is either from an original program feed to an insertion feed, from an insertion feed to another insertion feed, or from an insertion feed back to the original feed. The switch engine SHALL maintain context such that the original feed may be presented in error cases or as default behavior when the final insertion within a group is completed.

On receipt of a DPI trigger the implementation SHALL determine whether there is a switch instruction matching the trigger. To match a switch instruction with a trigger the implementation SHALL determine whether there is a SwitchInstruction with the switchID attribute equal to the value in the synchronized_event_id field of the trigger.

Insertions are often made in groups, as is the case when several ad spots make up an ad break. A switch instruction MAY indicate that if a previous insertion within a group is not made, then subsequent insertions should not take place, as might be the case if a service is selected after the first spot in an ad break. An instruction indicates that the entire group must be inserted when the insertionIndivisible attribute is non-zero, and indicates its position within a group via the groupSize and positionInGroup attributes. Where a matched instruction's insertionIndivisible attribute is non-zero, and all previous instructions within its group have not caused a switch, the implementation SHALL not perform a switch. Requirements defined below for performing switches assume this behavior and do not repeat these statements.

Where an instruction matches a trigger:

- Where the value in the synchronized_event_context field of the trigger indicates an L0 trigger the implementation SHALL perform an L0 switch.
- Where the value in the synchronized_event_context field of the trigger indicates an L1 operation the implementation SHALL perform an L1 switch.

For time based switches the implementation will process DPI timeline messages if present and maintain a metadata timeline for the current service. See 27.2.3 for details. When the point on the metadata timeline is reached that is equal to or greater than the value of the switchStartTime (if non-zero) of any switch instruction for the service, and the triggerSwitch flag of the switch instruction is zero, the implementation SHALL perform an L0 switch.

### 27.3.1 Requirements to perform an L0 switch

The implementation SHALL present the program specified by the switch instruction destination field or by the default rules defined for the end of an insertion. Note that an L0 DPI switch is not a service selection, that is, an L0 switch does not effect the lifecycle of any applications associated with the selected service of the original channel, while it may cause the loading and launching of applications associated with the newly presenting program.

### 27.3.2 Requirements to perform an L1 switch

L1 switches are signaled by a set of DPI triggers, one for each component that will be replaced, i.e. one for video, one for audio, etc. L1 triggers are identified as such by the synchronized_event_context field of the trigger, see 27.2.2.

When an L1 video trigger matching a switch instruction is received the implementation SHALL discontinue processing the video component of the currently presenting program and begin processing the video component of the program indicated by the destination field of the instruction.

When an L1 audio trigger matching a switch instruction is received the implementation SHALL discontinue processing the audio component of the currently presenting program and begin processing the first audio component indicated in the PMT of the program indicated by the destination field of the instruction, or the audio component indicated by the instructions audioChannel attribute if non-zero.

The latency between discontinuation of processing an original component and processing a destination component SHALL NOT be less than 1 millisecond and SHALL NOT exceed 30 milliseconds.

When a set of video and audio triggers has been processed, the implementation SHALL process all data streams associated with the destination program. This is for the purpose of launching applications that may be associated with the destination program. Note that the destination program may be the original program in the case where the switch is from an insertion feed back to the original feed.

### 27.3.3 Selecting a DPI enabled program

The following requirements for maintaining timelines and performing time-based switches when a service is selected are to accommodate scenarios in which a viewer tunes away from a program and tunes back during an insertion opportunity.

Where a metadata timeline has been established for a service, see 27.2.3, the implementation SHALL maintain the timeline if there is an instruction for the service with non-zero values for switchStartTime and switchEndTime and another service is selected. If a third service is selected, the implementation MAY discontinue maintaining the timeline for the original service. If the original service is not re-selected within an hour, the implementation MAY discontinue maintaining the timeline.

If a service is selected and its timeline is active, and the value of the timeline is greater than or equal the switchStartTime and less than or equal switchWindowEndTime, the implementation SHALL perform an L0 switch; an exception to this requirement is where switchWindowEndTime is within 4 seconds of the switchEndTime: the implementation SHALL not perform an L0 switch within 4 seconds of the switchEndTime.

### 27.3.4 Switch engine events

If a trigger is encountered and there is no corresponding instruction, an UNARMED_TRIGGER event SHALL be propagated to any listening targeting engine. If an insertion feed is currently being presented, the switch engine SHALL revert to presenting the original feed.

Where a switch occurs, an INSERTION_START event SHALL be propagated to any listening targeting engine.

If a viewer selects a DPI enabled service at a point between triggers within a set of L1 triggers, the receiver will not receive all of the triggers within the set, and therefore SHALL NOT perform a switch. Specifically, if the position_in_set field of a trigger is greater than or equal to two and a trigger has not been received on the service with position_in_set equal to 1, all triggers within the set SHALL be ignored. If not already presenting the original program, the implementation SHALL switch back to the original program and a MISSED_TRIGGER event SHALL be propagated to any listening targeting engine.

When a switch instruction expires, see Annex X, an INSTRUCTION_EXPIRED event SHALL be propagated to any targeting engine, and the instruction SHALL be considered inactive.

# Annex A (void)

# Annex B OCAP Specific Network APIs

The OpenCable Applications Platform requires specific network extensions to DVB-MHP in the following area:

DVB-J Return Channel Management API - OpenCable has a different set of return channel types.

The org.ocap.net package includes these extensions.

## B.1 DVB-GEM and DVB-MHP Specification Correspondence

This section of the OCAP 1.1 Profile corresponds to Annex R of [DVB-MHP1.1.2] and [DVB-GEM 1.1] as follows:

***Table B–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP 1.1 Section | [DVB-GEM 1.1] Section | [DVB-MHP1.1.2] Section | Compliance/ Deviation |
|---|---|---|---|
| Annex B OCAP Specific Network APIs | Annex R, DVB-J Return Channel Connection Management API | | Extension to GEM |
| Annex B.1 DVB-GEM and DVB-MHP Specification Correspondence | No corresponding section | | No corresponding section |
| Annex B.2 OCAP 1.1 Specific Requirements | No corresponding section | | No corresponding section |
| Annex B.2.1 Deviations to DVB-MHP | No corresponding section | | Extension to GEM |
| Annex B.2.1.1 Extensions to the DVB-J Return Channel Management API | getType() method | | Extension to GEM |
| Annex B.2.2 Extensions to DVB-MHP (Normative) | No corresponding section | | No corresponding section |
| Annex B.2.3 Java.net Usage | No corresponding section | | Extension to GEM |

This section corresponds to Annex R of the [DVB-MHP1.1.2]. The OpenCable Application Platform is in complete compliance with the API, `org.dvb.net.rc`, specified in this section.

## B.2 OCAP 1.1 Specific Requirements

### B.2.1 Deviations to DVB-MHP

The DVB-J Return Channel Management API has a broader scope than what is necessary for OCAP 1.1. In particular, DVB addresses CATV, PSTN/ISDN, DECT, GSM, LMDS, and SMATV networks. OCAP 1.1 needs to support CATV networks (specifically [SCTE 55-2] / [SCTE 55-1] and DOCSIS) only. To address these differences the DVB-J Return Channel Management APIs will be extended to return the OpenCable specific type of return channel present in the system.

[HOST 2.0] describes in detail the supported return channel interfaces that are available in OpenCable devices. This section details how the return channel interfaces are utilized in an OpenCable system and the APIs available to access them.

#### *B.2.1.1 Extensions to the DVB-J Return Channel Management API*

The org.dvb.net.rc.RCInterface.getType() SHALL return TYPE_CATV as the type of a return channel. The org.ocap.net.OCRCInterface.getSubType() method SHALL return either SUBTYPE_CATV_DOCSIS to indicate a DOCSIS return channel or SUBTYPE_CATV_OOB to indicate an OOB return channel that is specified in [SCTE 55-2] or [SCTE 55-1]. In case of SUBTYPE_CATV_OOB, the return channel can be accessed through the CableCARD/Host interface as specified in [CCIF 2.0].

### B.2.2 Extensions to DVB-MHP (Normative)

The following information is normative for the OCAP 1.1 Profile. It is specific to OCAP 1.1 and is not contained within the [DVB-MHP1.1.2].

### B.2.3 Java.net Usage

OCAP 1.1 provides two-way IP connectivity over the CableCARD/Host Extended Data Channel via the Out-of-Band channel (OOB-FDC/RDC) as defined in [CCIF 2.0]. An application may use java.net to access this communication channel.

# Annex C Permissions

This annex of the OCAP 1.1 Specification specifies APIs for Control Access permissions and Monitor Application permission.

In addition to the DVB APIs, this annex of the OCAP 1.1 Profile specifies APIs for privileged service information and selection permissions.

## C.1 DVB-GEM and DVB-MHP Specification Correspondence

This annex of the OCAP 1.1 Profile corresponds to Annex T of [DVB-MHP1.1.2] and [DVB-GEM 1.1] as follows:

***Table C–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| Annex C Permissions | Annex T (normative): Permissions | Compliance |
| Annex C.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension |
| Annex C.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension |
| Annex C.2.1 Extensions to DVB-MHP (Normative) | No Corresponding Section | OCAP-Specific Extension |
| Annex C.2.1.1 Monitor Application Permission | No Corresponding Section | OCAP-Specific Extension |

## C.2 OCAP 1.1 Specific Requirements

### C.2.1 Extensions to DVB-MHP (Normative)

This information extends the specification requirements made to the [DVB-MHP1.1.2].

The following information is normative for the OCAP 1.1 Profile. It is specific to OCAP 1.1 and is not contained within the [DVB-MHP1.1.2].

#### *C.2.1.1 Monitor Application Permission*

The Monitor Application permission provides multiple permissions that are intended for Monitor Applications or applications handling Monitor Application like functionality. The Monitor Application permission is signaled in the permission request file for each application; see Section 14.2.1.9. For usage details see Annex Q.

#### *C.2.1.2 Registered API User Permission*

The Registered API User permission provides permission for an application to make use of a specific API that has been registered through the Registered API Manager. The implementation shall grant this permission for each corresponding and valid entry in the permission request file for the application (unless denied by the then active SecurityPolicyHandler). For usage details, please see Annex Q

# Annex D HAVi Level 2 User Interface

This section is the Home Audio/Video Interoperability Architecture Level 2 User Interface. This HAVi user interface is designed as a TV-friendly user interface framework and requires elements from the Java Abstract Windowing Tool kit (AWT).

## D.1 DVB-GEM and DVB-MHP Specification Correspondence

This section of the OCAP 1.1 Profile complies with the HAVi specification referenced by [DVB-MHP1.1.2] as follows:

***Table D–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| Annex D HAVi Level 2 User Interface | VOID | Unknown | Annex V | Unknown |
| Annex D.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | Unknown | No Corresponding Section | Unknown |
| Annex D.2 OCAP 1.1 Specific Requirements | No Corresponding Section | Unknown | No Corresponding Section | Unknown |
| Annex D.2.1 Deviations from the HAVi Specification | No Corresponding Section | Unknown | No Corresponding Section | Unknown |
| Annex D.2.1.1 org.havi.ui.event.HRcCapabilities | No Corresponding Section | Unknown | No Corresponding Section | Unknown |
| Annex D.2.1.2 org.havi.ui.event.HEventRepresentation | No Corresponding Section | Unknown | No Corresponding Section | Unknown |
| Annex D.2.2 Extensions to DVB-MHP (Normative) | No Corresponding Section | Unknown | No Corresponding Section | Unknown |
| Annex D.2.2.1 OCAP 1.1 Extensions to HRcEvent | No Corresponding Section | Unknown | No Corresponding Section | Unknown |

The OpenCable Application Platform is in complete compliance with the HAVi API, `org.havi.ui`, as specified in this section.

## D.2 OCAP 1.1 Specific Requirements

### D.2.1 Deviations from the HAVi Specification

The OpenCable Application Platform is in compliance with the HAVi API, `org.havi.ui.event`, except where modifications have been noted below.

#### *D.2.1.1 org.havi.ui.event.HRcCapabilities*

The org.havi.ui.event.HRcCapabilities SHALL support OCAP key codes extended by Annex D.2.2.1.

#### *D.2.1.2 org.havi.ui.event.HEventRepresentation*

The org.havi.ui.event.HEventRepresentation SHALL support OCAP key codes extended by Annex D.2.2.1.

### D.2.2 Extensions to DVB-MHP (Normative)

This information extends the specification requirements made to the [DVB-MHP1.1.2].

The following information is normative for the OCAP 1.1 Profile. It is specific to OCAP 1.1 and is not contained within the [DVB-MHP1.1.2].

#### *D.2.2.1 OCAP 1.1 Extensions to HRcEvent*

The OpenCable Application Platform is adding the following event definitions to the HrcEvent class of [HAVi]. These event definitions are encapsulated by the OCRcEvent class located in the org.ocap.ui.event package. (See Annex E.)

public static final int VK_RF_BYPASS

The RF Bypass key code indicates a user request to bypass the Host video processing if Host device supports RF Bypass.

public static final int VK_MENU

The Menu key code indicates a user request to display a setup menu.

public static final int VK_EXIT

The Exit key code indicates a user request to terminate the guide or other displayed functions.

public static final int VK_LAST

The Last key code indicates a user request to tune to the previous program.

public static final int VK_NEXT_FAVORITE_CHANNEL

The Next Favorite Channel key code indicates a user request to tune to the next favorite channel in the list.

public static final int VK_ON_DEMAND

The on demand key code indicates a user request to access on demand functions.

# Annex E OCAP 1.1 UI Event API

This section presents the `org.ocap.ui` and `org.ocap.ui.event` APIs.

***Table E–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| Annex E OCAP 1.1 UI Event API | No Corresponding Section | OCAP-Specific Extension |

## Package org.ocap.ui

| Interface Summary | |
|---|---|
| **HSceneBinding** | Defines a binding between the area on a display that an HScene is (or will) occupy and the associated application |
| **HSceneChangeRequestHandler** | Interface to be implemented by a privileged application in order to handle requests to display an HScene not currently displayed, or change the positions of HScenes on the screen, or move an HScene in the 'z' order. |

| Class Summary | |
|---|---|
| **HSceneManager** | This class represents a manager that lets an application register a handler to requested HScene changes within a logical HScreen composited with all HScenes. |

### org.ocap.ui
### Interface HSceneBinding

```
public interface HSceneBinding
```

Defines a binding between the area on a display that an HScene is (or will) occupy and the associated application

| Method Summary | |
|---|---|
| `OcapAppAttributes` | **`getAppAttributes`**`()`<br>Gets the attributes of the application associated with the HScene. |
| `org.havi.ui.HScreenRectangle` | **`getRectangle`**`()`<br>Gets the rectangle in normalized co-ordinates that the HScene is using or has requested to use. |

# Method Detail

### getRectangle

`org.havi.ui.HScreenRectangle` **`getRectangle`**`()`

Gets the rectangle in normalized co-ordinates that the HScene is using or has requested to use.

**Returns:**

Rectangle of the HScene corresponding to the application attributes.

### getAppAttributes

`OcapAppAttributes` **`getAppAttributes`**`()`

Gets the attributes of the application associated with the HScene.

**org.ocap.ui**

## Interface HSceneChangeRequestHandler

```
public interface HSceneChangeRequestHandler
```

Interface to be implemented by a privileged application in order to handle requests to display an HScene not currently displayed, or change the positions of HScenes on the screen, or move an HScene in the 'z' order.

# Method Summary

| | |
|---|---|
| boolean | **testMove**(HSceneBinding move, HSceneBinding[] currentScenes)<br>Tests whether an HScene move request can be allowed or not. |
| boolean | **testOrder**(HSceneBinding[] currentScenes, int currentOrder, int newOrder)<br>Tests if an HScene z-order change request can be made or not. |
| boolean | **testShow**(HSceneBinding newScene, HSceneBinding[] oldScenes)<br>Tests whether an HScene display request can be allowed or not. |

# Method Detail

### testShow

```
boolean testShow(HSceneBinding newScene,
                 HSceneBinding[] oldScenes)
```

Tests whether an HScene display request can be allowed or not. The implementation SHALL call this method whenever the HScene is to be displayed including when the HScene show or setvisible(true) methods are called.

**Parameters:**

`newScene` - the new HScene to be displayed

`oldScenes` - the existing displayed HScenes

**Returns:**

true if the new HScene is allowed to be displayed false if it is not allowed to be displayed

### testMove

```
boolean testMove(HSceneBinding move,
                 HSceneBinding[] currentScenes)
```

Tests whether an HScene move request can be allowed or not. Called when an HScene is to be moved around the HScreen or resized.

**Parameters:**

`move` - the new location/size of the HScene.

`currentScenes` - the existing HScenes including the current location of the HScene to move.

**Returns:**

True if the move can be made, otherwise returns false.

#### testOrder

```
boolean testOrder(HSceneBinding[] currentScenes,
                  int currentOrder,
                  int newOrder)
```

Tests if an HScene z-order change request can be made or not. Called when an HScene is to be moved in z-order.

**Parameters:**

`currentScenes` - the existing displayed HScenes in z-order with entry 0 being the front.

`currentOrder` - the existing position in the currentScene array of the HScene to move.

`newOrder` - the new position that it is requested to move the HScene to.

**Returns:**

True if the move can be made, otherwise returns false.

## Package org.ocap.ui.event

| Class Summary | |
|---|---|
| **OCRcEvent** | The OCAP remote control event class. |

#### org.ocap.ui.event
### Class OCRcEvent

```
java.lang.Object
  └ java.util.EventObject
      └ java.awt.AWTEvent
          └ java.awt.event.ComponentEvent
              └ java.awt.event.InputEvent
                  └ java.awt.event.KeyEvent
                      └ org.havi.ui.event.HRcEvent
                          └ org.ocap.ui.event.OCRcEvent
```

**All Implemented Interfaces:**

java.io.Serializable

```
public class OCRcEvent
extends org.havi.ui.event.HRcEvent
```

The OCAP remote control event class. This class provides constants of key codes extended by OCAP.

The presence or absence of these keys and their desired representation is provided by the org.havi.ui.event.HRcCapabilities and the org.havi.ui.event.HEventRepresentaion.

Instances of OCRcEvent are reported through the normal java.awt event mechanism. Note that the reception of these events by a java.awt.Component is dependent on it having java.awt.event.KeyEvent events enabled.

Note that it is an implementation option if remote control key events are repeated. All KEY PRESSED, KEY TYPED and KEY RELEASED events shall be generated. An application is able to determine whether a key is being continuously pressed by containing logic to detect the KEY RELEASED event after a KEY PRESSED event.

| Field Summary | |
|---|---|
| static int | **OCRC_FIRST**<br>Marks the first integer id for the range of OCAP remote control key codes. |
| static int | **OCRC_LAST**<br>Marks the last integer id for the range of OCAP remote control key codes. |
| static int | **VK_APPS**<br>The 'apps' key code. |
| static int | **VK_BACK**<br>The 'back' key code. |
| static int | **VK_EXIT**<br>The 'exit' key code. |
| static int | **VK_FORWARD**<br>The 'forward' key code. |
| static int | **VK_LAST**<br>The 'last' key code. |
| static int | **VK_LINK**<br>The 'link' key code. |
| static int | **VK_LIST**<br>The ’list’ key code. |
| static int | **VK_LIVE**<br>The ’live’ key code. |
| static int | **VK_LOCK**<br>The ’lock’ key code. |
| static int | **VK_MENU**<br>The 'menu' key code. |
| static int | **VK_NEXT_DAY**<br>The guide 'next day' key code. |
| static int | **VK_NEXT_FAVORITE_CHANNEL**<br>The ’next favorite channel’ key code. |
| static int | **VK_ON_DEMAND**<br>The ’on demand’ key code. |
| static int | **VK_PINP_DOWN**<br>The ’picture-in-picture down’ key code. |
| static int | **VK_PINP_MOVE**<br>The ’picture-in-picture move’ key code. |
| static int | **VK_PINP_UP**<br>The ’picture-in-picture up’ key code. |
| static int | **VK_PREV_DAY**<br>The guide 'previous day' key code. |
| static int | **VK_RESERVE_1**<br>The ’reserved’ key code number 1. |
| static int | **VK_RESERVE_2**<br>The ’reserved’ key code number 2. |

| Field Summary | |
|---|---|
| static int | **VK_RESERVE_3**<br>The ’reserved’ key code number 3. |
| static int | **VK_RESERVE_4**<br>The ’reserved’ key code number 4. |
| static int | **VK_RESERVE_5**<br>The ’reserved’ key code number 5. |
| static int | **VK_RESERVE_6**<br>The ’reserved’ key code number 6. |
| static int | **VK_RF_BYPASS**<br>The 'RF Bypass' key code. |
| static int | **VK_SETTINGS**<br>The 'settings' key code. |
| static int | **VK_SKIP**<br>The ’skip’ key code. |
| static int | **VK_ZOOM**<br>The 'zoom' key code. |

| Fields inherited from class org.havi.ui.event.HRcEvent |
|---|
| RC_FIRST, RC_LAST, VK_BALANCE_LEFT, VK_BALANCE_RIGHT, VK_BASS_BOOST_DOWN, VK_BASS_BOOST_UP, VK_CHANNEL_DOWN, VK_CHANNEL_UP, VK_CLEAR_FAVORITE_0, VK_CLEAR_FAVORITE_1, VK_CLEAR_FAVORITE_2, VK_CLEAR_FAVORITE_3, VK_COLORED_KEY_0, VK_COLORED_KEY_1, VK_COLORED_KEY_2, VK_COLORED_KEY_3, VK_COLORED_KEY_4, VK_COLORED_KEY_5, VK_DIMMER, VK_DISPLAY_SWAP, VK_EJECT_TOGGLE, VK_FADER_FRONT, VK_FADER_REAR, VK_FAST_FWD, VK_GO_TO_END, VK_GO_TO_START, VK_GUIDE, VK_INFO, VK_MUTE, VK_PINP_TOGGLE, VK_PLAY, VK_PLAY_SPEED_DOWN, VK_PLAY_SPEED_RESET, VK_PLAY_SPEED_UP, VK_POWER, VK_RANDOM_TOGGLE, VK_RECALL_FAVORITE_0, VK_RECALL_FAVORITE_1, VK_RECALL_FAVORITE_2, VK_RECALL_FAVORITE_3, VK_RECORD, VK_RECORD_SPEED_NEXT, VK_REWIND, VK_SCAN_CHANNELS_TOGGLE, VK_SCREEN_MODE_NEXT, VK_SPLIT_SCREEN_TOGGLE, VK_STOP, VK_STORE_FAVORITE_0, VK_STORE_FAVORITE_1, VK_STORE_FAVORITE_2, VK_STORE_FAVORITE_3, VK_SUBTITLE, VK_SURROUND_MODE_NEXT, VK_TELETEXT, VK_TRACK_NEXT, VK_TRACK_PREV, VK_VIDEO_MODE_NEXT, VK_VOLUME_DOWN, VK_VOLUME_UP, VK_WINK |

| Fields inherited from class java.awt.event.KeyEvent |
| --- |
| CHAR_UNDEFINED, KEY_FIRST, KEY_LAST, KEY_LOCATION_LEFT, KEY_LOCATION_NUMPAD, KEY_LOCATION_RIGHT, KEY_LOCATION_STANDARD, KEY_LOCATION_UNKNOWN, KEY_PRESSED, KEY_RELEASED, KEY_TYPED, VK_0, VK_1, VK_2, VK_3, VK_4, VK_5, VK_6, VK_7, VK_8, VK_9, VK_A, VK_ACCEPT, VK_ADD, VK_AGAIN, VK_ALL_CANDIDATES, VK_ALPHANUMERIC, VK_ALT, VK_ALT_GRAPH, VK_AMPERSAND, VK_ASTERISK, VK_AT, VK_B, VK_BACK_QUOTE, VK_BACK_SLASH, VK_BACK_SPACE, VK_BEGIN, VK_BRACELEFT, VK_BRACERIGHT, VK_C, VK_CANCEL, VK_CAPS_LOCK, VK_CIRCUMFLEX, VK_CLEAR, VK_CLOSE_BRACKET, VK_CODE_INPUT, VK_COLON, VK_COMMA, VK_COMPOSE, VK_CONTEXT_MENU, VK_CONTROL, VK_CONVERT, VK_COPY, VK_CUT, VK_D, VK_DEAD_ABOVEDOT, VK_DEAD_ABOVERING, VK_DEAD_ACUTE, VK_DEAD_BREVE, VK_DEAD_CARON, VK_DEAD_CEDILLA, VK_DEAD_CIRCUMFLEX, VK_DEAD_DIAERESIS, VK_DEAD_DOUBLEACUTE, VK_DEAD_GRAVE, VK_DEAD_IOTA, VK_DEAD_MACRON, VK_DEAD_OGONEK, VK_DEAD_SEMIVOICED_SOUND, VK_DEAD_TILDE, VK_DEAD_VOICED_SOUND, VK_DECIMAL, VK_DELETE, VK_DIVIDE, VK_DOLLAR, VK_DOWN, VK_E, VK_END, VK_ENTER, VK_EQUALS, VK_ESCAPE, VK_EURO_SIGN, VK_EXCLAMATION_MARK, VK_F, VK_F1, VK_F10, VK_F11, VK_F12, VK_F13, VK_F14, VK_F15, VK_F16, VK_F17, VK_F18, VK_F19, VK_F2, VK_F20, VK_F21, VK_F22, VK_F23, VK_F24, VK_F3, VK_F4, VK_F5, VK_F6, VK_F7, VK_F8, VK_F9, VK_FINAL, VK_FIND, VK_FULL_WIDTH, VK_G, VK_GREATER, VK_H, VK_HALF_WIDTH, VK_HELP, VK_HIRAGANA, VK_HOME, VK_I, VK_INPUT_METHOD_ON_OFF, VK_INSERT, VK_INVERTED_EXCLAMATION_MARK, VK_J, VK_JAPANESE_HIRAGANA, VK_JAPANESE_KATAKANA, VK_JAPANESE_ROMAN, VK_K, VK_KANA, VK_KANA_LOCK, VK_KANJI, VK_KATAKANA, VK_KP_DOWN, VK_KP_LEFT, VK_KP_RIGHT, VK_KP_UP, VK_L, VK_LEFT, VK_LEFT_PARENTHESIS, VK_LESS, VK_M, VK_META, VK_MINUS, VK_MODECHANGE, VK_MULTIPLY, VK_N, VK_NONCONVERT, VK_NUM_LOCK, VK_NUMBER_SIGN, VK_NUMPAD0, VK_NUMPAD1, VK_NUMPAD2, VK_NUMPAD3, VK_NUMPAD4, VK_NUMPAD5, VK_NUMPAD6, VK_NUMPAD7, VK_NUMPAD8, VK_NUMPAD9, VK_O, VK_OPEN_BRACKET, VK_P, VK_PAGE_DOWN, VK_PAGE_UP, VK_PASTE, VK_PAUSE, VK_PERIOD, VK_PLUS, VK_PREVIOUS_CANDIDATE, VK_PRINTSCREEN, VK_PROPS, VK_Q, VK_QUOTE, VK_QUOTEDBL, VK_R, VK_RIGHT, VK_RIGHT_PARENTHESIS, VK_ROMAN_CHARACTERS, VK_S, VK_SCROLL_LOCK, VK_SEMICOLON, VK_SEPARATER, VK_SEPARATOR, VK_SHIFT, VK_SLASH, VK_SPACE, VK_SUBTRACT, VK_T, VK_TAB, VK_U, VK_UNDEFINED, VK_UNDERSCORE, VK_UNDO, VK_UP, VK_V, VK_W, VK_WINDOWS, VK_X, VK_Y, VK_Z |

| Fields inherited from class java.awt.event.InputEvent |
| --- |
| ALT_DOWN_MASK, ALT_GRAPH_DOWN_MASK, ALT_GRAPH_MASK, ALT_MASK, BUTTON1_DOWN_MASK, BUTTON1_MASK, BUTTON2_DOWN_MASK, BUTTON2_MASK, BUTTON3_DOWN_MASK, BUTTON3_MASK, CTRL_DOWN_MASK, CTRL_MASK, META_DOWN_MASK, META_MASK, SHIFT_DOWN_MASK, SHIFT_MASK |

| Fields inherited from class java.awt.event.ComponentEvent |
| --- |
| COMPONENT_FIRST, COMPONENT_HIDDEN, COMPONENT_LAST, COMPONENT_MOVED, COMPONENT_RESIZED, COMPONENT_SHOWN |

| Fields inherited from class java.awt.AWTEvent |
| --- |
| ACTION_EVENT_MASK, ADJUSTMENT_EVENT_MASK, COMPONENT_EVENT_MASK, consumed, CONTAINER_EVENT_MASK, FOCUS_EVENT_MASK, HIERARCHY_BOUNDS_EVENT_MASK, HIERARCHY_EVENT_MASK, id, INPUT_METHOD_EVENT_MASK, INVOCATION_EVENT_MASK, ITEM_EVENT_MASK, KEY_EVENT_MASK, MOUSE_EVENT_MASK, MOUSE_MOTION_EVENT_MASK, MOUSE_WHEEL_EVENT_MASK, PAINT_EVENT_MASK, RESERVED_ID_MAX, TEXT_EVENT_MASK, WINDOW_EVENT_MASK, WINDOW_FOCUS_EVENT_MASK, WINDOW_STATE_EVENT_MASK |

| Fields inherited from class java.util.EventObject |
| --- |
| source |

# Constructor Summary

**OCRcEvent**(java.awt.Component source, int id, long when, int modifiers, int keyCode, char keyChar)
Constructs an OCRcEvent object.

# Method Summary

| Methods inherited from class java.awt.event.KeyEvent |
| --- |
| getKeyChar, getKeyCode, getKeyLocation, getKeyModifiersText, getKeyText, isActionKey, paramString, setKeyChar, setKeyCode, setModifiers |

| Methods inherited from class java.awt.event.InputEvent |
| --- |
| consume, getModifiers, getModifiersEx, getModifiersExText, getWhen, isAltDown, isAltGraphDown, isConsumed, isControlDown, isMetaDown, isShiftDown |

| Methods inherited from class java.awt.event.ComponentEvent |
| --- |
| getComponent |

| Methods inherited from class java.awt.AWTEvent |
| --- |
| getID, setSource, toString |

| Methods inherited from class java.util.EventObject |
| --- |
| getSource |

| Methods inherited from class java.lang.Object |
| --- |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

# Field Detail

### OCRC_FIRST

```
public static final int OCRC_FIRST
```
Marks the first integer id for the range of OCAP remote control key codes.

### VK_RF_BYPASS

```
public static final int VK_RF_BYPASS
```
The 'RF Bypass' key code. Indicates a user request to toggle bypass on and off, if the host device supports RF bypass, by passing the RF input directly to the host RF output. This is only applicable to devices that have a channel 3/4 RF output.

### VK_EXIT

```
public static final int VK_EXIT
```
The 'exit' key code. Indicates a user request to exit the current application.

### VK_MENU

```
public static final int VK_MENU
```
The 'menu' key code. Indicates a user request for an on-screen menu (toggle).

### VK_NEXT_DAY

```
public static final int VK_NEXT_DAY
```
The guide 'next day' key code. Indicates a user request for the next day's worth of EPG data from the guide application.
**See Also:**
VK_PREV_DAY

### VK_PREV_DAY

```
public static final int VK_PREV_DAY
```
The guide 'previous day' key code. Indicates a user request for the previous day's worth of EPG data from the guide applications.
**See Also:**
VK_NEXT_DAY

### VK_APPS

```
public static final int VK_APPS
```
The 'apps' key code. Indicates a user request for applications.

### VK_LINK

`public static final int` **`VK_LINK`**

The 'link' key code. Indicates a user request for launching linked content.

#### VK_LAST

`public static final int` **`VK_LAST`**

The 'last' key code. Indicates a user request for tuning to the last channel tuned.

#### VK_BACK

`public static final int` **`VK_BACK`**

The 'back' key code. Indicates a user request moving to the previous URL or web page.

**See Also:**
VK_FORWARD

#### VK_FORWARD

`public static final int` **`VK_FORWARD`**

The 'forward' key code. Indicates a user request to move to the next URL or web page.

**See Also:**
VK_BACK

#### VK_ZOOM

`public static final int` **`VK_ZOOM`**

The 'zoom' key code. Indicates a user request to toggle from full-screen to scaled between TV and data pages.

#### VK_SETTINGS

`public static final int` **`VK_SETTINGS`**

The 'settings' key code. Indicates a user request to access the settings (user id, email account, parental control, etc.).

#### VK_NEXT_FAVORITE_CHANNEL

`public static final int` **`VK_NEXT_FAVORITE_CHANNEL`**

The ’next favorite channel’ key code. Indicates a user request to tune to the next favorite channel.

#### VK_RESERVE_1

`public static final int` **`VK_RESERVE_1`**

The ’reserved’ key code number 1. Reserved for future use.

#### VK_RESERVE_2

`public static final int` **`VK_RESERVE_2`**

The ’reserved’ key code number 2. Reserved for future use.

#### VK_RESERVE_3

```
public static final int VK_RESERVE_3
```
The 'reserved' key code number 3. Reserved for future use.

#### VK_RESERVE_4

```
public static final int VK_RESERVE_4
```
The 'reserved' key code number 4. Reserved for future use.

#### VK_RESERVE_5

```
public static final int VK_RESERVE_5
```
The 'reserved' key code number 5. Reserved for future use.

#### VK_RESERVE_6

```
public static final int VK_RESERVE_6
```
The 'reserved' key code number 6. Reserved for future use.

#### VK_LOCK

```
public static final int VK_LOCK
```
The 'lock' key code. Indicates a user request to lock the current program.

#### VK_SKIP

```
public static final int VK_SKIP
```
The 'skip' key code. Indicates a user request to skip the current program.

#### VK_LIST

```
public static final int VK_LIST
```
The 'list' key code. Indicates a user request to list the current program.

#### VK_LIVE

```
public static final int VK_LIVE
```
The 'live' key code. Indicates a user request to view live programs.

#### VK_ON_DEMAND

```
public static final int VK_ON_DEMAND
```
The 'on demand' key code. Indicates a user request to access on demand functions.

#### VK_PINP_MOVE

```
public static final int VK_PINP_MOVE
```
The 'picture-in-picture move' key code. Indicates a user request to move the picture-in-picture window.

#### VK_PINP_UP

```
public static final int VK_PINP_UP
```
The 'picture-in-picture up' key code. Indicates a user request to move the picture-in-picture window up.

#### VK_PINP_DOWN

```
public static final int VK_PINP_DOWN
```
The 'picture-in-picture down' key code. Indicates a user request to move the picture-in-picture window down.

#### OCRC_LAST

```
public static final int OCRC_LAST
```
Marks the last integer id for the range of OCAP remote control key codes.

## Constructor Detail

#### OCRcEvent

```
public OCRcEvent(java.awt.Component source,
                 int id,
                 long when,
                 int modifiers,
                 int keyCode,
                 char keyChar)
```
Constructs an `OCRcEvent` object.

**Parameters:**

`source` - the object where the event originated.
`id` - the identifier in the range KEY_FIRST to KEY_LAST.
`when` - the time stamp for this event.
`modifiers` - indication of any modification keys that are active for this event.
`keyCode` - the code of the key associated with this event.
`keyChar` - the character representation of the key associated with this event.

# Annex F OCAP 1.1 Hardware API

***Table F–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| Annex F OCAP 1.1 Hardware API | No Corresponding Section | OCAP-Specific Extension |

### Package org.ocap.hardware

| Interface Summary | |
|---|---|
| **IEEE1394Node** | This interface represents the information on a 1394 node. |
| **PowerModeChangeListener** | The callback interface to be implemented by classes that wish to receive notification when the power mode of the Host Devices changes (for example when the user presses the Power button). |

| Class Summary | |
|---|---|
| **CopyControl** | This class represents the copy control information on the analog and digital A/V outputs of the OCAP terminal. |
| **Host** | This class represents the host terminal device and provides access to the Host ID, raw image data, the power state of the host and a java.util.Enumeration of references to VideoOutputPort instances. |
| **VideoOutputPort** | An object of this class represents an analog or digital video output of the OCAP terminal. |

### org.ocap.hardware
## Class CopyControl

```
java.lang.Object
  └ org.ocap.hardware.CopyControl

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

This class represents the copy control information on the analog and digital A/V outputs of the OCAP terminal.

| Constructor Summary | |
|---|---|
| protected | **CopyControl**()<br>Do not use. |

# Method Summary

| | |
|---|---|
| `static int` | **`getCCIBits`**`(javax.tv.service.Service Service)`<br>Provides an OCAP Application with the ability to query the OpenCable Host Device for the current value of the CCI bits, which the OpenCable Host Device is currently using for Copy Protection. |

| **Methods inherited from class java.lang.Object** |
|---|
| `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait` |

# Constructor Detail

### CopyControl

`protected` **`CopyControl`**`()`

Do not use. This is to prevent a public constructor being generated.

# Method Detail

### getCCIBits

`public static final int` **`getCCIBits`**`(javax.tv.service.Service Service)`

Provides an OCAP Application with the ability to query the OpenCable Host Device for the current value of the CCI bits, which the OpenCable Host Device is currently using for Copy Protection. Note (informative) OCAP Applications that have access to and are processing video content should call this function at a periodic rate of no less than once every minute.

**Parameters:**

`Service` - indicates the service for which the returned CCI value applies to. CCI values are passed from a CableCARD to a Host associated with a program number. The implementation SHALL use the service to identify the program number.

**Returns:**

The CCI values currently in use by the OpenCable Host Device for the indicated service.

**Throws:**

`java.lang.IllegalStateException` - If the parameter Service is not presenting.

**org.ocap.hardware**

## Class Host

```
java.lang.Object
  └ org.ocap.hardware.Host
```

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

This class represents the host terminal device and provides access to the Host ID, raw image data, the power state of the host and a java.util.Enumeration of references to VideoOutputPort instances. See also org.ocap.OcapSystem to get the singleton instance.

## Field Summary

| | |
|---|---|
| static int | **FULL_POWER**<br>Power mode constant for normal "on" mode. |
| static int | **LOW_POWER**<br>Power mode constant for "standby" mode. |

## Constructor Summary

| | |
|---|---|
| protected | **Host()**<br>A constructor of this class. |

## Method Summary

| | |
|---|---|
| void | **addPowerModeChangeListener**(PowerModeChangeListener l)<br>Adds the PowerModeChangeListener to be called (PowerModeChangeListener.powerModeChanged(int) when the power mode of the box changes (for example when the user presses the Power button). |
| void | **codeDownload()**<br>This method initiates a download of the operating software in the Host as specified by the CableCARD Interface 2.0 Specification [4]. |
| boolean | **getACOutlet()**<br>Query whether power to the AC Outlet, if present, is currently On (true) or Off (false) NOTE: AC Outlet refers to an external power plug on the STB. |
| java.lang.String | **getID()**<br>Get a human-readable string representing the ID of this Host. |
| static Host | **getInstance()**<br>This method returns a singleton system-wide instance of the Host class. |
| int | **getPowerMode()** |

# Method Summary

| | |
|---|---|
| java.lang.String | **getReverseChannelMAC**()<br>Gets the MAC address used by the Host for reverse channel unicast communications. |
| boolean | **getRFBypass**()<br>Queries whether RF Bypass is currently enabled. |
| boolean | **getRFBypassCapability**()<br>Returns capability of RF bypass control on the host. |
| java.util.Enumeration | **getVideoOutputPorts**()<br>This method returns a java.util.Enumeration of references to VideoOutputPort instances. |
| boolean | **isACOutletPresent**()<br>Query whether there is an AC Outlet on the STB. |
| void | **reboot**()<br>This method initiates a reboot of the Host device. |
| void | **removePowerModeChangeListener**(PowerModeChangeListener l)<br>Removes the previously-added PowerModeChangeListener. |
| void | **setACOutlet**(boolean enable)<br>Switch power to AC Outlet, if present, On (true) or Off (false) NOTE: AC Outlet refers to an external power plug on the STB. |
| void | **setRFBypass**(boolean enable)<br>Enables or disables RF Bypass. |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Field Detail

### FULL_POWER

```
public static final int FULL_POWER
```
Power mode constant for normal "on" mode.

### LOW_POWER

```
public static final int LOW_POWER
```
Power mode constant for "standby" mode.

# Constructor Detail

### Host

`protected` **`Host`**`()`

A constructor of this class. An application must use the `getInstance()` method to create an instance.

# Method Detail

### getInstance

`public static Host` **`getInstance`**`()`

This method returns a singleton system-wide instance of the Host class.

**Returns:**

a singleton Host instance.

### getID

`public java.lang.String` **`getID`**`()`

Get a human-readable string representing the ID of this Host. This should be a string that could be read over the phone to an MSO that uniquely identifies the Host.

**Returns:**

id String host id

### getPowerMode

`public int` **`getPowerMode`**`()`

**Returns:**

the current power mode of the box (for example LOW_POWER).

**See Also:**

`FULL_POWER`, `LOW_POWER`

### getReverseChannelMAC

`public java.lang.String` **`getReverseChannelMAC`**`()`

Gets the MAC address used by the Host for reverse channel unicast communications. This value SHALL match the value the Host would use in DSG mode for a DHCP request.

**Returns:**

MAC address of the Host.

### addPowerModeChangeListener

`public void` **`addPowerModeChangeListener`**`(PowerModeChangeListener l)`

Adds the PowerModeChangeListener to be called (`PowerModeChangeListener.powerModeChanged(int)` when the power mode of the box changes (for example when the user presses the Power button).

**Parameters:**

`l` - is an instance implementing PowerModeChangeListener whose powerModeChanged method will be called when the power mode of the Host Device changes.

### removePowerModeChangeListener

`public void` **`removePowerModeChangeListener`**`(PowerModeChangeListener l)`

Removes the previously-added PowerModeChangeListener.
**Parameters:**
`l` - is the PowerModeChangeListener to disable. Does nothing if l was never added, has been removed, or is null.

#### getVideoOutputPorts

`public java.util.Enumeration` **`getVideoOutputPorts`**`()`

This method returns a java.util.Enumeration of references to VideoOutputPort instances.
**Returns:**
the java.util.Enumeration of references to VideoOutputPort instances.

#### reboot

`public void` **`reboot`**`()`

This method initiates a reboot of the Host device. The method caller shall have the MonitorAppPermission("reboot").

Note that the `SystemEventListener.notifyEvent(org.ocap.system.event.SystemEvent)` method shall be called before the initiated reboot is performed by the Host device. The monitor application can clean up resources in the SystemEventListener.notifyEvent method call. After the SystemEventListener.notifyEvent method call returns, the Host device continues the reboot, i.e., the boot process described in the *Boot Process* Section of this specification will be done.

**Throws:**
`java.lang.SecurityException` - if the caller does not have the MonitorAppPermission("reboot").

#### codeDownload

`public void` **`codeDownload`**`()`

This method initiates a download of the operating software in the Host as specified by the CableCARD Interface 2.0 Specification [4].

**Throws:**
`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("codeDownload").

#### isACOutletPresent

`public boolean` **`isACOutletPresent`**`()`

Query whether there is an AC Outlet on the STB. NOTE: AC Outlet refers to an external power plug on the STB. That is, a device such as a VCR can plug into the STB for power.
**Returns:**
true if there is an AC Outlet, else false.

#### getACOutlet

`public boolean` **`getACOutlet`**`()`

Query whether power to the AC Outlet, if present, is currently On (true) or Off (false) NOTE: AC Outlet refers to an external power plug on the STB. That is, a device such as a VCR can plug into the STB for power.

**Returns:**
The current AC Outlet status (false = Off, true = On).
**Throws:**
`java.lang.IllegalStateException` - if this method is called when there is no AC Outlet.

#### setACOutlet

`public void setACOutlet(boolean enable)`

Switch power to AC Outlet, if present, On (true) or Off (false) NOTE: AC Outlet refers to an external power plug on the STB. That is, a device such as a VCR can plug into the STB for power.
**Parameters:**
`enable` - The power setting for the AC Outlet.
**Throws:**
`java.lang.IllegalStateException` - if this method is called when there is no AC Outlet.

#### getRFBypassCapability

`public boolean getRFBypassCapability()`

Returns capability of RF bypass control on the host.
**Returns:**
true if the host can control RF bypass on/off, else false.

#### getRFBypass

`public boolean getRFBypass()`

Queries whether RF Bypass is currently enabled. If RF Bypass is enabled, the incoming RF signal is directly routed to the RF output port when the host is in a stand by mode, thereby totally bypassing the host.
**Returns:**
true if RF Bypass is currently enabled, else false. If the host doesn't support RF bypass, false returns.

#### setRFBypass

`public void setRFBypass(boolean enable)`

Enables or disables RF Bypass. If RF Bypass is enabled, the incoming RF signal is directly routed to the RF output port when the host is in a stand by mode, thereby totally bypassing the host.
**Parameters:**
`enable` - If true, RF Bypass will be enabled. Otherwise it will be disabled.
**Throws:**
`java.lang.IllegalStateException` - if the host doesn't support RF bypass.

**org.ocap.hardware**

## Interface IEEE1394Node

```
public interface IEEE1394Node
```

This interface represents the information on a 1394 node.

# Method Summary

| | |
|---|---|
| byte[] | **getEUI64**()<br>Returns the value of EUI-64 of the 1394 node. |
| java.lang.String | **getModelName**()<br>Returns the value of MODEL NAME TEXTUAL DESCRIPTOR of the 1394 node. |
| short[] | **getSubunitType**()<br>Returns the list of subunitTypes supported by the 1394 node. |
| java.lang.String | **getVendorName**()<br>Returns the value of VENDOR NAME TEXTUAL DESCRIPTOR of the 1394 node. |

# Method Detail

### getEUI64

```
byte[] getEUI64()
```

Returns the value of EUI-64 of the 1394 node. EUI-64 is defined in IEEE Std 1394-1995.

**Returns:**

an unsigned big endian 64-bits value of EUI-64 of the 1394 node.

**Throws:**

`java.lang.SecurityException` - if the caller has not been granted MonitorAppPermission("setVideoPort").

### getModelName

```
java.lang.String getModelName()
```

Returns the value of MODEL NAME TEXTUAL DESCRIPTOR of the 1394 node. MODEL NAME TEXTUAL DESCRIPTOR is defined in EIA-775-A.

**Returns:**

the value of MODEL NAME TEXTUAL DESCRIPTOR of the 1394 node. If the 1394 node does not have the MODEL NAME TEXTUAL DESCRIPTOR, null is returned.

**Throws:**

`java.lang.SecurityException` - if the caller has not been granted MonitorAppPermission("setVideoPort").

### getVendorName

`java.lang.String` **`getVendorName`**`()`

Returns the value of VENDOR NAME TEXTUAL DESCRIPTOR of the 1394 node. VENDOR NAME TEXTUAL DESCRIPTOR is defined in EIA-775-A.

**Returns:**

the value of VENDOR NAME TEXTUAL DESCRIPTOR of the 1394 node If the 1394 node does not have the VENDOR NAME TEXTUAL DESCRIPTOR, null is returned.

**Throws:**

`java.lang.SecurityException` - if the caller has not been granted MonitorAppPermission("setVideoPort").

### getSubunitType

`short[]` **`getSubunitType`**`()`

Returns the list of subunitTypes supported by the 1394 node.

**Returns:**

the list of subunitTypes supported by the 1394 node. The subunit type is defined in EIA-775-A.

**Throws:**

`java.lang.SecurityException` - if the caller has not been granted MonitorAppPermission("setVideoPort")

**org.ocap.hardware**

## Interface PowerModeChangeListener

**All Superinterfaces:**
java.util.EventListener

```
public interface PowerModeChangeListener
extends java.util.EventListener
```

The callback interface to be implemented by classes that wish to receive notification when the power mode of the Host Devices changes (for example when the user presses the Power button).

# Method Summary

| | |
|---|---|
| void | **powerModeChanged**(int newPowerMode)<br>Called when the power mode changes (for example from full to low power). |

# Method Detail

**powerModeChanged**

```
void powerModeChanged(int newPowerMode)
```

Called when the power mode changes (for example from full to low power).

**See Also:**
Host.FULL_POWER, Host.LOW_POWER

**org.ocap.hardware**

## Class VideoOutputPort

```
java.lang.Object
  └ org.ocap.hardware.VideoOutputPort

public abstract class VideoOutputPort
extends java.lang.Object
```

An object of this class represents an analog or digital video output of the OCAP terminal. If the type is the analog video output, it is mapped to a physical pin plug of the video output. If the type is the digital serial output, it is mapped to a physical pin plug of the video output. If the type is the digital bus output, it is mapped to a bus node that has several output ports. For example, if the type is the 1394 bus, the VideoOutputPort instance represents not the 1394 port (physical pin plug) but the 1394 node that has several 1394 ports.

An application cannot construct an instance of this class directly. Instead, the Host.getVideoOutputPorts() method is used to obtain a java.util.Enumeration of references to VideoOutputPort instances. The Enumeration.nextElement() method can be used to obtain references to individual VideoOutputPort instances.

The video port is a scarce resource, but the resource management framework is not applied. At most only one 1394 connection is available for a single OCAP implementation. If some applications call VideoOutputPort.select1394sink(), only the last call is effective. The other calls are ignored or disconnected without any notification.

## Field Summary

| | |
|---|---|
| static int | **AV_OUTPUT_PORT_TYPE_1394**<br>AV Output Port Type 1394 (Firewire) |
| static int | **AV_OUTPUT_PORT_TYPE_BB**<br>AV Output Port Type Baseband (RCA connector) |
| static int | **AV_OUTPUT_PORT_TYPE_COMPONENT_VIDEO**<br>AV Output Port Type Component Video |
| static int | **AV_OUTPUT_PORT_TYPE_DVI**<br>AV Output Port Type DVI (Panel Link, HDCP) |
| static int | **AV_OUTPUT_PORT_TYPE_RF**<br>AV Output Port Type RF channel 3/4 |
| static int | **AV_OUTPUT_PORT_TYPE_SVIDEO**<br>AV Output Port Type S-Video |
| static int | **CAPABILITY_TYPE_DTCP**<br>AV Output Port Capability Type DTCP |
| static int | **CAPABILITY_TYPE_HDCP**<br>AV Output Port Capability Type HDCP |
| static int | **CAPABILITY_TYPE_RESOLUTION_RESTRICTION**<br>AV Output Port Capability Type Resolution Restriction for HD Video |

## Constructor Summary

| | |
|---|---|
| protected | **VideoOutputPort**()<br>OCAP applications SHALL NOT use this constructor - it is provided for internal use by the OCAP implementation. |

## Method Summary

| | |
|---|---|
| abstract void | **disable**()<br>Disable the video output port, that is, prevent the video output from this port. |
| abstract void | **enable**()<br>Enable the video output port, that is, allow the video output from this port. |
| abstract IEEE1394Node[] | **getIEEE1394Node**()<br>Get the list of IEEE1394Node corresponding to all the 1394 nodes that were discovered by the OpenCable Host device. |
| abstract int | **getType**()<br>Get the type of this VideoOutputPort. |
| abstract java.lang.Object | **queryCapability**(int capabilityType)<br>Query the value related to specified capabilityType. |
| abstract void | **selectIEEE1394Sink**(byte[] eui64, short subunitType)<br>Select an IEEE1394 sink node which will establish a stream connection to the node of the OCAP implementation and give a parameter to establish a point to point AV connection. |
| abstract boolean | **status**()<br>This method returns a current status of this video output port. |

| **Methods inherited from class java.lang.Object** |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

## Field Detail

### AV_OUTPUT_PORT_TYPE_RF

public static final int **AV_OUTPUT_PORT_TYPE_RF**

AV Output Port Type RF channel 3/4

### AV_OUTPUT_PORT_TYPE_BB

public static final int **AV_OUTPUT_PORT_TYPE_BB**

AV Output Port Type Baseband (RCA connector)

#### AV_OUTPUT_PORT_TYPE_SVIDEO

`public static final int` **`AV_OUTPUT_PORT_TYPE_SVIDEO`**
AV Output Port Type S-Video

#### AV_OUTPUT_PORT_TYPE_1394

`public static final int` **`AV_OUTPUT_PORT_TYPE_1394`**
AV Output Port Type 1394 (Firewire)

#### AV_OUTPUT_PORT_TYPE_DVI

`public static final int` **`AV_OUTPUT_PORT_TYPE_DVI`**
AV Output Port Type DVI (Panel Link, HDCP)

#### AV_OUTPUT_PORT_TYPE_COMPONENT_VIDEO

`public static final int` **`AV_OUTPUT_PORT_TYPE_COMPONENT_VIDEO`**
AV Output Port Type Component Video

#### CAPABILITY_TYPE_DTCP

`public static final int` **`CAPABILITY_TYPE_DTCP`**
AV Output Port Capability Type DTCP

#### CAPABILITY_TYPE_HDCP

`public static final int` **`CAPABILITY_TYPE_HDCP`**
AV Output Port Capability Type HDCP

#### CAPABILITY_TYPE_RESOLUTION_RESTRICTION

`public static final int` **`CAPABILITY_TYPE_RESOLUTION_RESTRICTION`**
AV Output Port Capability Type Resolution Restriction for HD Video

## Constructor Detail

#### VideoOutputPort

`protected` **`VideoOutputPort`**`()`
OCAP applications SHALL NOT use this constructor - it is provided for internal use by the OCAP implementation. The result of calling this method from an application is undefined, and valid implementations MAY throw any Error or RuntimeException.

# Method Detail

### enable

`public abstract void` **`enable`**`()`

Enable the video output port, that is, allow the video output from this port. A stream connection is established and an AV stream is output. The status() method is used to confirm the result of this method call.

**Throws:**

`java.lang.SecurityException` - if the caller has not been granted MonitorAppPermission("setVideoPort")

`java.lang.IllegalStateException` - if the host couldn't enable the port in cases where the Host is unable to start a signal from the port, e.g. in the case where another 1394 port has a connection that prevents a new connection.

### disable

`public abstract void` **`disable`**`()`

Disable the video output port, that is, prevent the video output from this port. The stream connection is disconnected. If the port does not support a disabling function, this method affects nothing. The status() method is used to confirm the result of this method call. Note that the specific port types that support disabling are specified elsewhere, for example, by the OpenCable Host Core Functional Requirement, CHILA/PHILA. Note that FCC may provide rules for port disabling.

**Throws:**

`java.lang.SecurityException` - if the caller has not been granted MonitorAppPermission("setVideoPort")

`java.lang.IllegalStateException` - if this method is called for a VideoOutputPort which does not support disabling, or the host couldn't disable the port in cases where the Host is unable to terminate a signal from the port, e.g. in the case where a 1394 port has overlayed connections.

### status

`public abstract boolean` **`status`**`()`

This method returns a current status of this video output port.

**Returns:**

enable/disable status of video output port. If true output port is enabled, otherwise it is disabled.

### queryCapability

`public abstract java.lang.Object` **`queryCapability`**`(int capabilityType)`

Query the value related to specified capabilityType.

**Parameters:**

`capabilityType` - The capability type to query the value CAPABILITY_TYPE_xxx

**Returns:**

The value related to the specified capabilityType will return as follows:

- CAPABILITY_TYPE_DTCP-java.lang.Boolean which indicates DTCP is available (TRUE) or not will return.
- CAPABILITY_TYPE_HDCP-java.lang.Boolean which indicates HDCP is available (TRUE) or not will return.

- CAPABILITY_TYPE_RESOLUTION_RESTRICTION-java.lang.Integer which indicates the restricted pixel size for HD video will return.

### getIEEE1394Node

```
public abstract IEEE1394Node[] getIEEE1394Node()
```

Get the list of IEEE1394Node corresponding to all the 1394 nodes that were discovered by the OpenCable Host device. The 1394 node which does not have EUI-64 is ignored.

**Returns:**

An array of IEEE1394Node. The first IEEE1394Node in the array represents the 1394 node of the OCAP implementation itself.

**Throws:**

`java.lang.IllegalStateException` - if this method is called for the VideoOutputPort which does not represent AV_OUTPUT_PORT_TYPE_1394.

`java.lang.SecurityException` - if the caller has not been granted MonitorAppPermission("setVideoPort")

### selectIEEE1394Sink

```
public abstract void selectIEEE1394Sink(byte[] eui64,
                                        short subunitType)
```

Select an IEEE1394 sink node which will establish a stream connection to the node of the OCAP implementation and give a parameter to establish a point to point AV connection. This method neither establishes a connection nor outputs a stream. An application must call VideoOutputPort.enable() to establish a connection and output a stream. The stream connection parameters which are not specified by this method are assigned by the OCAP implementation automatically. For example, oPCR is selected by the OCAP implementation. A source of an AV stream is a tuner of the OCAP implementation.

**Parameters:**

`eui64` - an unsigned big endian 64-bits value of EUI-64 of a sink node.

`subunitType` - type value of a sink AV subunit to be connected.

**Throws:**

`java.lang.IllegalArgumentException` - if eui64 is not valid.

`java.lang.IllegalStateException` - if this method is called for the VideoOutputPort which does not represent AV_OUTPUT_PORT_TYPE_1394.

`java.lang.SecurityException` - if the caller has not been granted MonitorAppPermission("setVideoPort")

### getType

```
public abstract int getType()
```

Get the type of this VideoOutputPort.

**Returns:**

The integer representation of the VideoOutputPort type. That is, one of the AV_OUTPUT_PORT_TYPE constants.

# Annex G OCAP 1.1 Application API

This section presents the `org.ocap.application` APIs.

***Table G–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| Annex G OCAP 1.1 Application API | No Corresponding Section | OCAP-Specific Extension |

## Package org.ocap.application

This package contains APIs for controlling the lifecycle of applications.

| Interface Summary | |
|---|---|
| **AppFilterHandler** | Application programs can provide an implementation of this interface to an `AppFilter` to make part of decision for `AppFilter.accept`. |
| **AppSignalHandler** | A class implementing this interface can interrupt updating application information in the AppsDatabase caused by XAIT acquisition. |
| **OcapAppAttributes** | This interface represents various information about an application registered in the AppsDatabase. |
| **SecurityPolicyHandler** | This interface provides a callback handler to modify the Permissions granted to an application to be launched. |

| Class Summary | |
|---|---|
| **AppFilter** | *AppFilter* provides a means of filtering AppIDs. |
| **ApplicationMode** | Defines the set of available modes for an application. |
| **AppManagerProxy** | This class represents the application manager functionality used by the Monitor Application. |
| **AppPattern** | AppPattern is an element that constitutes an `AppFilter`. |
| **PermissionInformation** | This class contains information to allow the monitor application to choose the permissions to grant to an application. |

## Package org.ocap.application Description

This package contains APIs for controlling the lifecycle of applications. This package is primarily used by the monitor application.

## Application Registration

This package provides methods for registering and unregistering unbound applications. The AppManagerProxy.registerUnboundApp(InputStream) method registers unbound applications specified by the InputStream that contains XAIT data. This is a similar function described as the signaling of unbound application using the XAIT (see section 11.2.2.1*).* The AppManagerProxy.unregisterUnboundApp(AppID) method unregisters an unbound application specified by AppID.

The format of the XAIT SHALL follow Section 11.2.2.3 OCAP 1.0 XAIT stated in this Specification.

This is sample code of registering and unregistering an unbound application.

```
import org.ocap.application.*;
import org.dvb.application.*;
import java.io.*;


public class MARegistering {
    /**
     * Constructor of this class.
     */
    public MARegistering(byte [] xaitData) {

        AppManagerProxy appMgrProxy = AppManagerProxy.getInstance();

        /* register unbound applications. */
        try {
            ByteArrayInputStream xait =
                    new ByteArrayInputStream(xaitData);
            appMgrProxy.registerUnboundApp(xait);
        }
        catch (IOException e) {
        }

        /* unregister an unbound application that is registered by a    */
        /* sample XAIT defined by the xaitData.                         */
        AppID appid = new AppID(0x1234, 0xABCD);
        appMgrProxy.unregisterUnboundApp(appid);
    }
}
```

## Application Information

The org.ocap.application.OcapAppAttributes shall be used instead of the org.dvb.application.AppAttributes.

## XAIT updating Management

In order to manage the updating of network signaled applications on the receiver, the monitor application MAY reject a new XAIT to abort updating unbound application information in the AppsDatabase. It can set the AppSignalHandler via AppManagerProxy.setAppSignalHandler(AppSignalHandler). When a new XAIT is received, the AppSignalHandler.notifyXAITUpdate(OcapAppAttributes[]) is called to allow the monitor application to make a decision of whether to update unbound application information.

This is sample code to abort updating unbound application information.

```
This is sample code to abort updating unbound application information.
```

```
import org.ocap.application.*;
import org.dvb.application.*;
import java.security.*;
import java.util.*;


public class MAXaitUpdater implements AppSignalHandler {
    /**
     * Constructor of this class.
     */
    public MAXaitUpdater() {
        AppManagerProxy appMgrProxy = AppManagerProxy.getInstance();

        /* Register AppSignalHandler. */
        appMgrProxy.setAppSignalHandler(this);
    }

    /**
     * Implement the notifyXAITUpdate method defined in the AppSignalHandler.
     */
    public boolean notifyXAITUpdate(OcapAppAttributes[] newApps) {
        /* Investigate the signaled applications. */
        AppID appid = newApps[0].getIdentifier();
        int controlCode = newApps[0].getApplicationControlCode();

        /* Write a code to make decision here. */

        /* The monitor application abort the updating of this XAIT. */
        return false;
    }
}
```

## Policy and Security Management

Black and white list support is provided by the AppFilter class. The application manager allows a filter to be set which all applications must pass through before being run.

The monitor application can register an application filter to prevent applications from running. When an application is being launched, the application manager tests the application against the filter. If the test fails, the application will be blocked as described in Chapter 21. See `AppManagerProxy.setAppFilter` method for filter registration.

This is sample code of application filtering. The monitor application MAY create a unique application filter class that extends the `org.dvb.application.AppsDatabaseFilter` class. It MAY implement an unique algorithm to filter an application in the accept() method.

```
import org.ocap.application.*;
import org.dvb.application.*;

public class MAAppFilter extends AppsDatabaseFilter {
    /**
     * Constructor of this class.
     */
    public MAAppFilter() {
        AppManagerProxy appMgrProxy = AppManagerProxy.getInstance();

        /* Register an application filter. */
        appMgrProxy.setAppFilter(this);
    }
```

```
    /**
     * Implement the accept() method defined in the AppsDatabaseFilter.
     */
    public boolean accept(AppID appid) {
        int REJECTED_OID = 0x1234;

        /* Investigate the specified applications. */
        if(appid.getOID() == REJECTED_OID) {
            return false;
        }
        return true;
    }
}
```

The monitor application MAY set the SecurityPolicyHandler via the AppManagerProxy.setSecurityPolicyHandler(SecurityPolicyHandler) method. For those applications that pass through the current application filter, the SecurityPolicyHandler.getAppPermissions(PermissionsInformation) method is called. The monitor application can get a PermissionCollection and return it as the return value of getAppPermissions method. The application is launched using the modified PermissionCollection.

Monitor applications that set the SecurityPolicyHandler should take care when setting permissions for Host Device Manufacturer applications (i.e. applications where PermissionCollection.isManufacturerApp() returns true). Denying permissions to Host Device Manufacturer applications may cause an extremely poor user experience.

This is sample code for modifying PermissionCollection. It denies the AppsControlPermission to a specified application, but grants all other requested permissions.

```
import org.ocap.application.*;
import org.dvb.application.*;
import java.security.*;
import java.util.*;


public class MAPermissionModifier implements SecurityPolicyHandler {
    /**
     * Constructor of this class.
     */
    public MAPermissionModifier() {
        AppManagerProxy appMgrProxy = AppManagerProxy.getInstance();

        /* Register SecurityPolicyHandler applications. */
        appMgrProxy.setSecurityPolicyHandler(this);
    }

    /**
     * Implement the getAppPermission method defined in SecurityPolicyHandler.
     */
    public PermissionCollection getAppPermissions(
            PermissionInformation permissionInfo) {

        /* Investigate the requested PermissionCollection here. */
        AppID appid = permissionInfo.getAppID();
        PermissionCollection requestedPermissionCollection
                = permissionInfo.getRequestedPermissions();

        /* Give manufacturer applications everything they ask for */
        if (permissionInfo.isManufacturerApp()) {
            return requestedPermissionCollection;
```

```
        }

        /* Start with the basic permissions for unsigned applications */
        /* Note that we are guaranteed that these permissions will always */
        /* be a subset of the requested permissions */
        Permissions newPermissionCollection = new Permissions();
        Enumeration e =
PermissionInformation.getUnsignedAppPermissions().elements();
        while (e.hasMoreElements()) {
            newPermissionCollection.add((Permission)e.nextElement())
        }


        /* The permission we are going to deny */
        Permission appsControlPermission = new AppsControlPermission();
        AppId denyAppsControlPermissionAppId = new AppId(1, 2);

        /* Modify the PermissionCollection here. */
        /* Note that the modified permissions shall be a subset of the */
        /* requested permission. */
        e = requestedPermissionCollection.elements();
        while (e.hasMoreElements()) {
            Permission requested = (Permission)e.nextElement();
            if (!newPermissionCollection.implies(requested)) {
                /* It's not a permission we have already granted.  Test it. */
                /* (The above test is an optimization to avoid granting */
                /* the unsigned app permissions twice) */
                if (requested.implies(appsControlPermission)
                 && appid.equals(denyAppsControlPermissionAppId)) {
                     /* Deny requested permission */
                } else {
                     /* ... could have other tests here ... */
                     /* Grant requested permission */
                    newPermissionCollection.add(requested);
                }
            }
        }


        return newPermissionCollection;
    }
}
```

**org.ocap.application**
## Class AppFilter

```
java.lang.Object
  └ org.dvb.application.AppsDatabaseFilter
      └ org.ocap.application.AppFilter

public class AppFilter
extends org.dvb.application.AppsDatabaseFilter
```

*AppFilter* provides a means of filtering AppIDs. As a subclass of `AppsDatabaseFilter`, the method `accept(org.dvb.application.AppID)` makes a `true`/`false` decision based on an AppID.

An AppFilter contains a list of zero or more `AppPattern`s. Each AppPattern has the attributes: *pattern*, *action*, and *priority*. *pattern* specifies a group of AppIDs with a pair of ranges for organization ID and application ID. *action* specifies an action assigned to the AppID group; either `AppPattern.ALLOW`, `AppPattern.DENY`, or `AppPattern.ASK`. *priority* specifies this AppPattern's position in the search order: the biggest number comes first. Applications can insert an AppPattern anywhere in the search order by using the priority attribute effectively (`AppFilter.add`). When two or more AppPatterns in an AppFilter have the same priority, the search order among them is undefined. It is not recommendable to use AppPatterns that have the same priority but different actions.

When `accept` is called, the given AppID is compared to the AppID group of each AppPattern in the search order until a match is found. Then, it returns `true` or `false` if the action of matching AppPattern is `ALLOW` or `DENY` respectively. If no match is found, `accept` returns `true`.

If the action of matching AppPattern is `ASK`, then AppFilter calls `AppFilterHandler.accept` for the final decision; the matching AppPattern is handed over to this method. Applications can specify the `AppFilterHandler` with `AppFilter.setAskHandler`. If no AppFilterHandler is set, AppFilter returns `true`.

AppPatterns can have an expiration time and MSO-private information (*expirationTime* and *info*). `accept` and `getAppPatterns` methods ignore AppPatterns that have expired. The implementation may delete expired AppPatterns from AppFilter.

Example:

```
import org.ocap.application.*;
import org.dvb.application.AppID;

AppManagerProxy am = ...;
AppPattern[] patterns = {
    /* note that search order is dictated by "priority" */
    new AppPattern("10-5f:1-ff", AppPattern.ALLOW, 40),     // #3
    new AppPattern("30:2c-34", AppPattern.ALLOW, 100),      // #1
    new AppPattern("20-40", AppPattern.DENY, 80),           // #2
};
AppFilter af = new AppFilter(patterns);

/* false - matches "20-40" */
boolean badOne = af.accept(new AppID(0x30, 0x10));

/* true - matches "30:2c-34" */
```

```
boolean goodOne = af.accept(new AppID(0x30, 0x30));

/* will be the second entry: priority between 100 and 80 */
af.add(new AppPattern("40-4f:1000-1fff", DENY, 90));

/* register af with the system */
am.setAppFilter(af);
```

**See Also:**

`AppPattern`, `AppFilterHandler`, `AppManagerProxy`, `AppID`, `AppsDatabaseFilter`

# Constructor Summary

| |
|---|
| **`AppFilter`**`()`<br>Constructs an empty AppFilter. |
| **`AppFilter`**`(AppPattern[] patterns)`<br>Constructs an AppFilter with initial AppPatterns. |

# Method Summary

| | |
|---|---|
| `boolean` | **`accept`**`(org.dvb.application.AppID appID)`<br>Returns whether this AppFilter accepts the given AppID. |
| `void` | **`add`**`(AppPattern pattern)`<br>Adds an AppPattern to this AppFilter. |
| `java.util.Enumeration` | **`getAppPatterns`**`()`<br>Returns the AppPatterns in this AppFilter. |
| `boolean` | **`remove`**`(AppPattern pattern)`<br>Removes an AppPattern that equals to `pattern` in this AppFilter. |
| `void` | **`setAskHandler`**`(AppFilterHandler handler)`<br>Sets the handler to call when `accept` hits an AppPatterns with action `AppPattern.ASK`. |

| **Methods inherited from class java.lang.Object** |
|---|
| `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait` |

# Constructor Detail

### AppFilter

`public` **`AppFilter`**`()`

Constructs an empty AppFilter.

### AppFilter

`public` **`AppFilter`**`(AppPattern[] patterns)`

Constructs an AppFilter with initial AppPatterns.

**Parameters:**

`patterns` - AppPatterns to constitute an AppFilter.

# Method Detail

### getAppPatterns

`public java.util.Enumeration` **`getAppPatterns`**`()`

Returns the AppPatterns in this AppFilter.

**Returns:**

the enumeration of AppPatterns. When this AppFilter has no AppPattern, this method returns an empty Enumeration, not `null`.

### accept

`public boolean` **`accept`**`(org.dvb.application.AppID appID)`

Returns whether this AppFilter accepts the given AppID.

**Specified by:**

`accept` in class `org.dvb.application.AppsDatabaseFilter`

**Parameters:**

`appID` - an AppID to test.

**Returns:**

`true` if `appID` passes this filter.

### add

`public void` **`add`**`(AppPattern pattern)`

Adds an AppPattern to this AppFilter.

**Parameters:**

`pattern` - the AppPattern to add

### remove

`public boolean` **`remove`**`(AppPattern pattern)`

Removes an AppPattern that equals to `pattern` in this AppFilter. If this AppFilter does not contain `pattern`, it is unchanged.

**Parameters:**

`pattern` - the AppPattern to remove.

**Returns:**

`true` if the AppFilter contained the specified AppPattern.

**See Also:**

`AppPattern.equals(java.lang.Object)`

### setAskHandler

`public void` **`setAskHandler`**`(AppFilterHandler handler)`

Sets the handler to call when `accept` hits an AppPatterns with action `AppPattern.ASK`.

If a handler is already registered with this AppFilter, the new handler replaces it.

**Parameters:**
`handler` - the handler to set.

**org.ocap.application**
## Interface AppFilterHandler

```
public interface AppFilterHandler
```

Application programs can provide an implementation of this interface to an `AppFilter` to make part of decision for `AppFilter.accept`.

**See Also:**
`AppFilter.setAskHandler(org.ocap.application.AppFilterHandler)`

# Method Summary

| | |
|---|---|
| `boolean` | **`accept`**`(org.dvb.application.AppID appID, AppPattern matchingItem)`<br>This method is called by `AppFilter.accept(org.dvb.application.AppID)` when it finds a matching `AppPattern` whose action is ASK. |

# Method Detail

**accept**

```
boolean accept(org.dvb.application.AppID appID,
               AppPattern matchingItem)
```

This method is called by `AppFilter.accept(org.dvb.application.AppID)` when it finds a matching `AppPattern` whose action is ASK.

The return value of this method will be the return value of `AppFilter.accept`. The semantics of this method is identical to `AppsDatabaseFilter.accept(org.dvb.application.AppID)` except that the additional parameter `matchingItem` could be used as a hint.

**Parameters:**
`appID` - an AppID to test.
`matchingItem` - the `AppPattern` in `AppFilter` that matched `appID`
**Returns:**
`true` if `appID` passes this filter.

**org.ocap.application**
## Class ApplicationMode

```
java.lang.Object
  └ org.ocap.application.ApplicationMode
```

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

Defines the set of available modes for an application.

# Field Summary

| | |
|---|---|
| static ApplicationMode | **BACKGROUND**<br>The application is in background mode. |
| static ApplicationMode | **CROSSENVIRONMENT**<br>The application is in cross-environment mode. |
| static ApplicationMode | **NORMAL**<br>The application is in normal mode. |
| static ApplicationMode | **PAUSED**<br>The application is in paused mode. |

# Constructor Summary

| | |
|---|---|
| protected | **ApplicationMode**(java.lang.String s)<br>This protected constructor is provided to enable the set of modes to be extended. |

# Method Summary

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Field Detail

**NORMAL**

```
public static final ApplicationMode NORMAL
```
The application is in normal mode.

#### CROSSENVIRONMENT

`public static final ApplicationMode` **`CROSSENVIRONMENT`**

The application is in cross-environment mode.

#### PAUSED

`public static final ApplicationMode` **`PAUSED`**

The application is in paused mode.

#### BACKGROUND

`public static final ApplicationMode` **`BACKGROUND`**

The application is in background mode.

## Constructor Detail

#### ApplicationMode

`protected` **`ApplicationMode`**`(java.lang.String s)`

This protected constructor is provided to enable the set of modes to be extended. It is not intended to be used by applications.

**org.ocap.application**
## Class AppManagerProxy

```
java.lang.Object
  └ org.ocap.application.AppManagerProxy
```

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

This class represents the application manager functionality used by the Monitor Application. It provides a means of acquiring application signaling and registering a new unbound application for applications that have MonitorAppPermission.

An application which has MonitorAppPermission may have a subclass of the AppsDatabaseFilter class, a class implementing the AppSignalHandler interface or a class implementing SecurityPolicyHandler interface and may set an instance of them in the AppManagerProxy.

See Section 10 Application Model and 11 Application Signaling in this specification for details.

## Constructor Summary

| | |
|---|---|
| protected | **AppManagerProxy**()<br>This is a constructor of this class. |

## Method Summary

| | |
|---|---|
| void | **enableTestApplications**(boolean flag)<br>Controls whether applications signaled with test_application_flag set to one are ignored (as normal). |
| static AppManagerProxy | **getInstance**()<br>This method returns the sole instance of the AppManagerProxy class. |
| OcapAppAttributes[] | **getQuarantinedApps**()<br>Return the list of applications quarantined by the implementation for broken or non-responsive behavior. |
| static int[] | **getSupportedApplicationTypes**()<br>This method returns the set of application types supported by this OCAP implementation. |
| boolean | **queryTestApplicationsEnabled**()<br>Returns whether applications signaled with test_application_flag set to one are ignored. |
| void | **registerUnboundApp**(java.io.InputStream xait)<br>This method registers new unbound application entries. |
| void | **removeQuarantinedApplication**(org.dvb.application.AppID app, long version)<br>Remove an application from the list of applications quarantined by the implementation. |

# Method Summary

| | |
|---|---|
| void | **setAppFilter**(org.dvb.application.AppsDatabaseFilter filter)<br>This method sets an instance of a concrete class that extends AppsDatabaseFilter that decides whether the application is allowed to be launched or not for all applications to be launched. |
| void | **setApplicationPriority**(int priority, org.dvb.application.AppID appId)<br>This method sets the priority for the application. |
| void | **setAppSignalHandler**(AppSignalHandler handler)<br>This method sets an instance of a class that implements the AppSignalHandler interface. |
| void | **setSecurityPolicyHandler**(SecurityPolicyHandler handler)<br>This method sets an instance of a class that implements the SecurityPolicyHandler interface. |
| void | **unregisterUnboundApp**(int serviceId, org.dvb.application.AppID appid)<br>This method unregisters an unbound application from the AppsDatabase. |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Constructor Detail

### AppManagerProxy

`protected` **`AppManagerProxy`**`()`

This is a constructor of this class. An application must use the `getInstance()` method to create an instance.

# Method Detail

### getInstance

`public static AppManagerProxy` **`getInstance`**`()`

This method returns the sole instance of the AppManagerProxy class. The AppManagerProxy instance is either a singleton for each OCAP application, or a singleton for an entire OCAP implementation.

**Returns:**

The AppManagerProxy instance.

### setAppFilter

`public void` **`setAppFilter`**`(org.dvb.application.AppsDatabaseFilter filter)`

This method sets an instance of a concrete class that extends AppsDatabaseFilter that decides whether the application is allowed to be launched or not for all applications to be launched. At most, only one instance

of a concrete class that extends AppsDatabaseFilter can be set to the AppManagerProxy. Multiple calls of this method replace the previous instance by a new one. If no AppsDatabaseFilter has been set, then any application is allowed to be launched. By default, no AppsDatabaseFilter is set, i.e., all applications are allowed to be launched. Note that the specified AppsDatabaseFilter can't prevent registering applications to a service.

**Parameters:**

`filter` - An instance of a concrete class of the AppsDatabaseFilter that decides whether the application is allowed to be launched or not. If null is set, the AppsDatabaseFilter will be removed.

**Throws:**

`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("handler.appFilter")

### setAppSignalHandler

`public void` **`setAppSignalHandler`**`(AppSignalHandler handler)`

This method sets an instance of a class that implements the AppSignalHandler interface. At most, only one AppSignalHandler can be set. Multiple calls of this method replace the previous instance by a new one. If no AppSignalHandler has been set, then application information is updated immediately. By default, no AppSignalHandler is set, i.e., application information is updated immediately. The OCAP implementation SHALL call the accept() method of the application filter whenever it launches any type of application. After the monitor application has indicated that it has set its filters, the accept() method in the monitor application is called prior to launching any application. If the method returns "false" for the application to be launched, the application MUST NOT be launched, otherwise the implementation continues with the process of launching the application. As an optimization, the implementation SHOULD mark, in the applications database, any applications that have not been accepted through filter. This mark SHOULD remain until the current filters are replaced and SHOULD be used to prevent repeated requests being sent to the filtering application to validate the launching of any application that has previously been denied. The implementation MUST remove all filtering marks in the case that the current filters are replaced or removed. If the application is not marked, as previously filtered out, the implementation MUST call any registered AppDatabaseFilter.accept() between the time that the implementation receives a request to launch, and the actual launch of the application.

**Parameters:**

`handler` - An instance of a class implementing the AppSignalHandler interface that decides whether application information is updated using the new version of the XAIT or not. If null is set, the AppSignalHandler be removed.

**Throws:**

`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("handler.registrar")

### setSecurityPolicyHandler

`public void` **`setSecurityPolicyHandler`**`(SecurityPolicyHandler handler)`

This method sets an instance of a class that implements the SecurityPolicyHandler interface. At most, only one SecurityPolicyHandler can be set. Multiple calls of this method replace the previous instance by a new one. If no SecurityPolicyHandler has been set, then the requested set of Permissions are granted to an application to be launched. By default, no SecurityPolicyHandler is set, i.e., the requested set of Permissions are granted to an application to be launched.

**Parameters:**

`handler` - An instance of a class implementing the SecurityPolicyHandler interface that may modify a set of Permission granted to an application to be launched. If null is set, the SecurityPolicyHandler is removed.

**Throws:**

`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("security")

### registerUnboundApp

```
public void registerUnboundApp(java.io.InputStream xait)
                        throws java.io.IOException
```

This method registers new unbound application entries.

Generally, the ServiceList and, for currently selected abstract services, the AppsDatabase are updated when a new XAIT is received from the network. This method registers new unbound application entries without the network signaled XAIT.

If there has already been an entry in the ServiceList or AppsDatabase registered via the registerUnboundApp(AppID) method with the same combination of an AppID and a service name, the existing entry is replaced by the one specified by this method. Errors in the xait are not indicated to the calling application and incorrect xait information will be treated as described in DVB MHP 1.1.2 Section 10.4.1 Data Errors.

Note that the application entry registered by this method is processed in the same manner as applications signaled by the XAIT. Note that the `AppSignalHandler.notifyXAITUpdate(org.ocap.application.OcapAppAttributes[])` method is not called.

**Parameters:**
`xait` - An instance of java.io.InputStream that provides an XAIT formatted stream. If an XAIT consists of multiple sections, an instance of java.io.InputStream SHALL provide simple concatenation of them. All section header values of each section shall be valid. Sections shall be concatenated in order of ascending section number. Duplicate sections should not be included. If duplicates are included in the java.io.InputStream this method will discard all but the first occurrence.
**Throws:**
`java.lang.IllegalArgumentException` - if the InputStream does not represent a sequence of XAIT sections with valid section headers. Note that this exception is not thrown when descriptors in the XAIT are invalid. The descriptors may be analyzed asynchronously.
`java.io.IOException` - if an I/O error occurs.
`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("registrar")

### unregisterUnboundApp

```
public void unregisterUnboundApp(int serviceId,
                                 org.dvb.application.AppID appid)
```

This method unregisters an unbound application from the AppsDatabase.

This method unregisters an existing unbound application entry from AppsDatabase without the XAIT signaling. The unbound application entry is specified by an appid and a service_id. The application must have been previously registered by a call to registerUnboundApp(InputStream) from the same application that is making the call to unregisterUnboundApp(AppID).

If there is no specified entry in the service at the time this method is called, this method has no effect. If the application to be unregistered has been launched, it shall be killed.

**Parameters:**
`serviceId` - The service identifier to which this application is registered.
`appid` - An AppID instance identifies the application entry to be unregistered from the service.
**Throws:**
`java.lang.IllegalArgumentException` - if this method attempts to modify an application signaled in the XAIT or an AIT or a host device manufacturer application.
`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("registrar")

#### getSupportedApplicationTypes

`public static int[] **getSupportedApplicationTypes**()`

This method returns the set of application types supported by this OCAP implementation. The values returned shall be those used in the application_type field of the AIT and XAIT.

**Returns:**

an array containing all the supported application types

#### setApplicationPriority

```
public void setApplicationPriority(int priority,
                                   org.dvb.application.AppID appId)
```

This method sets the priority for the application. This method can be called at any time but has no affect upon resource contention resolutions that occurred before it was called. The priority set SHALL persist until a new version of the application is signaled, a reboot occurs, or this method is called again If the application is running the priority will be change when the application is relaunched.

**Parameters:**

`priority` - New priority for the application with appId.

`appId` - Application identifier of the application to have its priority changed.

**Throws:**

`java.lang.SecurityException` - is thrown when the caller does not have MonitorAppPermission("servicemanager").

`java.lang.IllegalStateException` - if the application, i.e. Xlet, is in the active state, or is currently set at monitor application priority.

#### enableTestApplications

`public void **enableTestApplications**(boolean flag)`

Controls whether applications signaled with test_application_flag set to one are ignored (as normal). Calling this method specifying true as the parameter SHALL cause the implementation to ignore the value of test_application_flag in the signaling and behave as if the signaled value was zero. In the absence of a subsequent call to this method, the value set shall persist until the cable environment is re- * initialized.

**Parameters:**

`flag` - true if test applications are to be enabled, false if they are to be disabled

**Throws:**

`java.lang.SecurityException` - when the caller does not have MonitorAppPermission("testApplications").

#### queryTestApplicationsEnabled

`public boolean **queryTestApplicationsEnabled**()`

Returns whether applications signaled with test_application_flag set to one are ignored.

**Returns:**

false if applications signaled with test_application_flag set to one are ignored, true if test applications are enabled

#### getQuarantinedApps

`public OcapAppAttributes[] **getQuarantinedApps**()`

Return the list of applications quarantined by the implementation for broken or non-responsive behavior. If no applications have been quarantined or the implementation does not quarantine applications then an array of length zero SHALL be returned.

**Returns:**

OcapAppAttributes a list of applications

### removeQuarantinedApplication

```
public void removeQuarantinedApplication(org.dvb.application.AppID app,
                                         long version)
```

Remove an application from the list of applications quarantined by the implementation. To remove applications signaled with an unbound application descriptor, the version parameter SHALL match the version number of the entry in the quarantine list for that application. To remove applications signaled without a version number, -1 must be specified for the version.

**Parameters:**

`app` - the identifier of the application to remove

`version` - the version number of the application to remove or -1 for applications with no signaled version number

**Throws:**

`SecurityPermission` - if the calling application does not have MonitorAppPermission("quarantine")

**org.ocap.application**
## Class AppPattern

```
java.lang.Object
  └ org.ocap.application.AppPattern
```

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

AppPattern is an element that constitutes an `AppFilter`. An AppPattern has the following attributes:

- *idPattern* - a group of AppIDs.
- *action* - an action (ALLOW, DENY, or ASK) for matching applications.
- *priority* - a priority that determines the search order position in an AppFilter. The highest priority is 255, the lowest is 0.
- *expirationTime* - An expiration time. Optional.
- *info* - an MSO-private data. Optional. Could be a String. `AppFilterHandler` may use it for making a decision.

`idPattern` specifies an AppID group with a String: a pair of ranges for Organization IDs and Application IDs. The syntax is:

`"oid1[-oid2][:aid1[-aid2]]"`

- `oid1` and `oid2` specify a range of Organization IDs inclusive. Each of them must be a 32-bit value.
- `aid1` and `aid2` specify a range of Application IDs inclusive. Each of them must be a 16-bit value.
- `oid2` and `aid2` must be greater than `oid1` and `aid1`, respectively.
- The encoding of these IDs follows *14.5 Text encoding of application identifiers* of *DVB-MHP 1.1.2 [11]*; hexadecimal, lower case, no leading zeros.
- Symbols in brackets are optional.
- When `oid2` is omitted, only `oid1` is in the range.
- When `aid2` is omitted, only `aid1` is in the range.
- When both `aid1` and `aid2` are omitted, all Application IDs are in the range.

See `AppFilter` for the examples.

**See Also:**

`AppFilter`, `AppFilterHandler`

## Field Summary

| | |
|---|---|
| static int | **ALLOW**<br>When `AppFilter.accept` finds a matching `AppPattern` with this action, it returns `true`. |
| static int | **ASK**<br>When `AppFilter.accept` finds a matching `AppPattern` with this action, it asks `AppFilterHandler.accept` for the decision. |
| static int | **DENY**<br>When `AppFilter.accept` finds a matching `AppPattern` with this action, it returns `false`. |

## Constructor Summary

| |
|---|
| **AppPattern**(java.lang.String idPattern, int action, int priority)<br>Constructs a new AppPattern with no expiration. |
| **AppPattern**(java.lang.String idPattern, int action, int priority, java.util.Date expirationTime, java.lang.Object info)<br>Constructs a new AppPattern with an expiration time and MSO private information. |

## Method Summary

| | |
|---|---|
| boolean | **equals**(java.lang.Object that)<br>Indicates whether some other object is "equal to" this one. |
| int | **getAction**()<br>Returns the action associated with this AppPattern. |
| java.lang.String | **getAppIDPattern**()<br>Returns the pattern string that specifies a group of AppIDs. |
| java.util.Date | **getExpirationTime**()<br>Returns the time for this AppPattern to expire or `null` if it never expires. |
| int | **getPriority**()<br>Returns the search order priority of this AppPattern. |
| java.lang.Object | **getPrivateInfo**()<br>Returns MSO-private information of this AppPattern. |
| int | **hashCode**() |

| **Methods inherited from class java.lang.Object** |
|---|
| clone, finalize, getClass, notify, notifyAll, toString, wait, wait, wait |

# Field Detail

### ALLOW

```
public static final int ALLOW
```

When `AppFilter.accept` finds a matching `AppPattern` with this action, it returns `true`.

**See Also:**
`AppFilter.accept(org.dvb.application.AppID)`

### DENY

```
public static final int DENY
```

When `AppFilter.accept` finds a matching `AppPattern` with this action, it returns `false`.

**See Also:**
`AppFilter.accept(org.dvb.application.AppID)`

### ASK

```
public static final int ASK
```

When `AppFilter.accept` finds a matching `AppPattern` with this action, it asks `AppFilterHandler.accept` for the decision.

**See Also:**
`AppFilter.accept(org.dvb.application.AppID)`, `AppFilterHandler.accept(org.dvb.application.AppID, org.ocap.application.AppPattern)`

# Constructor Detail

### AppPattern

```
public AppPattern(java.lang.String idPattern,
                  int action,
                  int priority)
```

Constructs a new AppPattern with no expiration.

**Parameters:**
`idPattern` - a String to specify an AppID group.
`action` - an action.
`priority` - a search order priority.

**Throws:**
`java.lang.IllegalArgumentException` - `idPattern` has a bad format, `action` or `priority` is out of range.

### AppPattern

```
public AppPattern(java.lang.String idPattern,
                  int action,
                  int priority,
                  java.util.Date expirationTime,
                  java.lang.Object info)
```

Constructs a new AppPattern with an expiration time and MSO private information.

**Parameters:**
`idPattern` - a String to specify an AppID group.
`action` - an action.
`priority` - a search order priority.
`expirationTime` - time for this AppPattern to expire. `null` it never expires.
`info` - MSO specific information. Can be `null`.
**Throws:**
`java.lang.IllegalArgumentException` - `idPattern` has a bad format, `action` or `priority` is out of range.

# Method Detail

### getAppIDPattern

`public java.lang.String` **`getAppIDPattern`**`()`

Returns the pattern string that specifies a group of AppIDs.
**Returns:**
the pattern string.

### getAction

`public int` **`getAction`**`()`

Returns the action associated with this AppPattern.
**Returns:**
the action.

### getPriority

`public int` **`getPriority`**`()`

Returns the search order priority of this AppPattern.
**Returns:**
the search order priority.

### getExpirationTime

`public java.util.Date` **`getExpirationTime`**`()`

Returns the time for this AppPattern to expire or `null` if it never expires.
**Returns:**
the expiration time or `null`.

### getPrivateInfo

`public java.lang.Object` **`getPrivateInfo`**`()`

Returns MSO-private information of this AppPattern.
**Returns:**
the MSO private information.

### equals

`public boolean` **`equals`**`(java.lang.Object that)`

Indicates whether some other object is "equal to" this one.

This method does not factor in `expirationTime` or `info` attributes, but does compare `idPattern`, `action`, and `priority` attributes.

**Overrides:**
`equals` in class `java.lang.Object`

### hashCode

`public int` **`hashCode`**`()`

**Overrides:**
`hashCode` in class `java.lang.Object`

**org.ocap.application**

## Interface AppSignalHandler

```
public interface AppSignalHandler
```

A class implementing this interface can interrupt updating application information in the AppsDatabase caused by XAIT acquisition.

An application which has a MonitorAppPermission("registrar") may have a class implementing this interface, and may set an instance of it in the AppManagerProxy.

Whenever a new version of the XAIT is received by the OCAP implementation, the `notifyXAITUpdate(org.ocap.application.OcapAppAttributes[])` method shall be called before updating application information in the AppsDatabase. If the method returns true or if no AppSignalHandler is set, the OCAP implementation shall update application information using the new version of the XAIT and update the stored applications as defined by the storage priorities in new XAIT. Otherwise the OCAP implementation shall ignore the new XAIT.

# Method Summary

| | |
|---|---|
| boolean | **notifyXAITUpdate**(OcapAppAttributes[] newApps)<br>This is a callback method to inquire of an application which has MonitorAppPermission("registrar") whether the AppDatabase shall update current application information using a new version of the XAIT or not. |

# Method Detail

**notifyXAITUpdate**

```
boolean notifyXAITUpdate(OcapAppAttributes[] newApps)
```

This is a callback method to inquire of an application which has MonitorAppPermission("registrar") whether the AppDatabase shall update current application information using a new version of the XAIT or not.

**Parameters:**

`newApps` - A list of instances of the OCAPAppAttributes class associated with all the applications whose details are listed in the new version of the XAIT that is received by the OCAP implementation.

**Returns:**

true if the AppsDatabase shall update entire application information using the new version of XAIT immediately. false if the AppDatabase shall keep application information as it is.

**org.ocap.application**

## Interface OcapAppAttributes

**All Superinterfaces:**
org.dvb.application.AppAttributes

```
public interface OcapAppAttributes
extends org.dvb.application.AppAttributes
```

This interface represents various information about an application registered in the AppsDatabase. This interface extends the org.dvb.application.AppAttributes in the points of following:
Defining the OCAP Application types.
Adding the getControlFlag method to get the application_control_code flag as signaled in an AIT or an XAIT.

For applications which are signaled in the AIT or the XAIT, the mapping between the values returned by methods in this interface and the fields and descriptors of the AIT or the XAIT shall be as specified in this specification. Instance of the class implementing this interface are immutable.

org.dvb.application.AppsDatabase MUST return an instance of OcapAppAttributes by the getAppAttributes methods.

For this version of the OCAP profile, the getProfiles() method always returns "ocap.profile.basic_profile" and the getIsServiceBound() method returns true for unbound applications.

# Field Summary

| | |
|---|---|
| static int | **AUTOSTART**<br>This represents the application control code "AUTOSTART" defined for the application_control_code in an AIT or a XAIT. |
| static int | **DESTROY**<br>This represents the application control code "DESTROY" defined for the application_control_code in an AIT or a XAIT. |
| static int | **KILL**<br>This represents the application control code "KILL" defined for the application_control_code in an AIT or a XAIT. |
| static int | **OCAP_J**<br>The OCAP registered value for all OCAP-J applications. |
| static int | **PREFETCH**<br>This represents the application control code "PREFETCH" defined for the application_control_code in an AIT. |
| static int | **PRESENT**<br>This represents the application control code "PRESENT" defined for the application_control_code in an AIT or a XAIT. |
| static int | **REMOTE**<br>This represents the application control code "REMOTE" defined for the application_control_code in an AIT. |

| Fields inherited from interface org.dvb.application.AppAttributes |
|---|
| DVB_HTML_application, DVB_J_application |

# Method Summary

| | |
|---|---|
| boolean | **canRunAsMode**(ApplicationMode mode)<br>Tests whether this application is signaled as being able to run in the provided mode |
| int | **getApplicationControlCode**()<br>This method returns the application_control_code of the application represented by this interface. |
| ApplicationMode | **getApplicationMode**()<br>Return the current mode for this application. |
| int | **getStoragePriority**()<br>This method returns the currently set storage priority for the application. |
| long | **getVersionNumber**()<br>Return the version number of the application signaled. |
| boolean | **hasNewVersion**()<br>Indicates that a new version of the application is stored and this version will replace the currently launched version when a new lifecycle for this application starts. |

| Methods inherited from interface org.dvb.application.AppAttributes |
|---|
| getAppIcon, getIdentifier, getIsServiceBound, getName, getName, getNames, getPriority, getProfiles, getProperty, getServiceLocator, getType, getVersions, isStartable, isVisible |

# Field Detail

### OCAP_J

static final int **OCAP_J**

The OCAP registered value for all OCAP-J applications.

### AUTOSTART

static final int **AUTOSTART**

This represents the application control code "AUTOSTART" defined for the application_control_code in an AIT or a XAIT.

### PRESENT

static final int **PRESENT**

This represents the application control code "PRESENT" defined for the application_control_code in an AIT or a XAIT.

#### DESTROY

`static final int` **`DESTROY`**

This represents the application control code "DESTROY" defined for the application_control_code in an AIT or a XAIT.

#### KILL

`static final int` **`KILL`**

This represents the application control code "KILL" defined for the application_control_code in an AIT or a XAIT.

#### PREFETCH

`static final int` **`PREFETCH`**

This represents the application control code "PREFETCH" defined for the application_control_code in an AIT.

#### REMOTE

`static final int` **`REMOTE`**

This represents the application control code "REMOTE" defined for the application_control_code in an AIT.

## Method Detail

#### getApplicationControlCode

`int` **`getApplicationControlCode`**`()`

This method returns the application_control_code of the application represented by this interface.

**Returns:**

int The application_control_code of the application represented by this interface.

#### getStoragePriority

`int` **`getStoragePriority`**`()`

This method returns the currently set storage priority for the application.

**Returns:**

int The storage priority for a currently stored application or zero if the application is not stored.

#### hasNewVersion

`boolean` **`hasNewVersion`**`()`

Indicates that a new version of the application is stored and this version will replace the currently launched version when a new lifecycle for this application starts.

**Returns:**

True, if the currently launched application will be replaced by a new version when the next application lifecycle starts. False, if the application is not currently launched, or if the application is not currently stored, or if the stored version of the application matches the version that is currently launched.

#### getVersionNumber

`long` **`getVersionNumber`**`()`

Return the version number of the application signaled. If the application is signaled with a version number (e.g. an OCAP unbound application descriptor or an MHP 1.1 application storage descriptor) then the version number from that descriptor SHALL be returned. Otherwise -1 shall be returned.

**Returns:**

the version number of the application or -1

#### getApplicationMode

`ApplicationMode` **`getApplicationMode`**`()`

Return the current mode for this application.

**Returns:**

the current mode for this application

#### canRunAsMode

`boolean` **`canRunAsMode`**`(ApplicationMode mode)`

Tests whether this application is signaled as being able to run in the provided mode

**Parameters:**

`mode` - an application mode

**Returns:**

true if and only if the application is signaled as being able to run in the specified mode

**org.ocap.application**

## Class PermissionInformation

```
java.lang.Object
  └ org.ocap.application.PermissionInformation
```

```
public abstract class PermissionInformation
extends java.lang.Object
```

This class contains information to allow the monitor application to choose the permissions to grant to an application.

**See Also:**
`SecurityPolicyHandler`

## Constructor Summary

| | |
|---|---|
| protected | **PermissionInformation**()<br>OCAP applications SHALL NOT use this constructor - it is provided for internal use by the OCAP implementation. |

## Method Summary

| | |
|---|---|
| abstract org.dvb.application.AppID | **getAppID**()<br>This method returns an AppID of an application to be granted a requested set of Permissions that is returned by the `getRequestedPermissions()` method. |
| abstract java.security.cert.Certificate[][] | **getCertificates**()<br>Returns the set of valid certificates that were used to sign the application identified by the AppID returned by the getAppID() method. |
| abstract java.security.PermissionCollection | **getRequestedPermissions**()<br>This method returns the requested set of Permissions for the application specified by the AppID that is returned by the `getAppID()` method. |
| static java.security.PermissionCollection | **getUnsignedAppPermissions**()<br>This method returns the set of Permissions that are requested by all unsigned applications. |
| abstract boolean | **isManufacturerApp**()<br>Returns true if and only if the application identified by the AppID returned by the getAppID() is a Host Device Manufacturer applications. |
| abstract boolean | **isPrivilegedCertificate**(java.security.cert.Certificate cert)<br>Verifies that an end-entity certificate used to validate and application or file is a member of the list of privileged certificates in the privileged certificate descriptor. |

| Methods inherited from class java.lang.Object |
| --- |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Constructor Detail

### PermissionInformation

`protected` **`PermissionInformation`**`()`

OCAP applications SHALL NOT use this constructor - it is provided for internal use by the OCAP implementation. The result of calling this method from an application is undefined, and valid implementations MAY throw any Error or RuntimeException.

# Method Detail

### getAppID

`public abstract org.dvb.application.AppID` **`getAppID`**`()`

This method returns an AppID of an application to be granted a requested set of Permissions that is returned by the `getRequestedPermissions()` method.

**Returns:**

The AppID instance of an application to be granted a requested set of Permissions which is returned by the `getRequestedPermissions()` method.

**Throws:**

`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("security").

### isManufacturerApp

`public abstract boolean` **`isManufacturerApp`**`()`

Returns true if and only if the application identified by the AppID returned by the getAppID() is a Host Device Manufacturer applications.

**Returns:**

true if and only if the application identified by the AppID returned by the getAppID() is a Host Device Manufacturer application.

### getCertificates

`public abstract java.security.cert.Certificate[][]` **`getCertificates`**`()`

Returns the set of valid certificates that were used to sign the application identified by the AppID returned by the getAppID() method.

Note that for Host Device Manufacturer applications, this may be an empty array.

For unsigned applications, this shall be an empty array.

**Returns:**

The return value is a two dimensional array of certificates where each member of the outer dimension represents a certificate chain that authenticates the application. The order of certificate chains in the outer array is unspecified. Each member of the inner dimension contains a certificate in the chain with the root

certificate in the first member and the end-entity certificate in the final member of the array. Each certificate in the inner array authenticates the certificate contained in the next array member.

#### isPrivilegedCertificate

```
public abstract boolean
isPrivilegedCertificate(java.security.cert.Certificate cert)
```

Verifies that an end-entity certificate used to validate and application or file is a member of the list of privileged certificates in the privileged certificate descriptor.

**Parameters:**

`cert` - The X.509 certificate that is to be checked against the list of privileged certificates in the privileged certificate descriptor.

**Returns:**

The return value is set to true if the SHA-1 hash of the supplied certificate matches one of the hash values listed in the privileged certificate descriptor.

#### getUnsignedAppPermissions

```
public static java.security.PermissionCollection getUnsignedAppPermissions()
```

This method returns the set of Permissions that are requested by all unsigned applications. The contents of this set of permissions is defined elsewhere in this specification.

**Returns:**

A read-only instance of a sub class of PermissionCollection containing the set of Permissions for an unsigned application.

#### getRequestedPermissions

```
public abstract java.security.PermissionCollection getRequestedPermissions()
```

This method returns the requested set of Permissions for the application specified by the AppID that is returned by the `getAppID()` method.

For Host Device Manufacturer applications, this is the set of permissions requested for the application by the Host Device Manufacturer. Note that this may include manufacturer-specific permissions (e.g. a manufacturer-specific permission to access a DVD player API).

For other applications, the requested set of Permissions consists of Permissions that are requested in a permission request file and Permissions requested for unsigned applications.

Note that the requested set of Permissions always includes the permissions requested for unsigned applications, as returned by getUnsignedAppPermissions().

**Returns:**

An instance of a sub class of the PermissionCollection containing the requested set of Permissions for an application to be launched. The application is specified by the AppID returned by the `getAppID()` method.

**org.ocap.application**

## Interface SecurityPolicyHandler

```
public interface SecurityPolicyHandler
```

This interface provides a callback handler to modify the Permissions granted to an application to be launched. An application that has a MonitorAppPermission("security") can have a concrete class that implements this interface and set an instance of it to the AppManagerProxy.

The `getAppPermissions(org.ocap.application.PermissionInformation)` method shall be called before the OCAP implementation launches any type of application (e.g. before class loading of any OCAP-J application). The application shall then be loaded and started with the set of Permissions that are returned as the return value of this method.

**See Also:**
`AppManagerProxy.setSecurityPolicyHandler(org.ocap.application.SecurityPolicyHandler)`

# Method Summary

| | |
|---|---|
| `java.security.PermissionCollection` | **`getAppPermissions`**`(PermissionInformation permissionInfo)`<br>This callback method is used to modify the set of Permissions that is granted to an application to be launched. |

# Method Detail

### getAppPermissions

```
java.security.PermissionCollection
getAppPermissions(PermissionInformation permissionInfo)
```

This callback method is used to modify the set of Permissions that is granted to an application to be launched.

The OCAP implementation shall call this method before class loading of any application, if an instance of a class that implements the `SecurityPolicyHandler` interface is set to the `AppManagerProxy`. The permissionInfo parameter of this method contains the AppID of the application to be launched and a requested set of Permissions that consists of Permissions requested in a permission request file and Permissions requested for the unsigned application. This method can modify the requested set of Permissions and returns them as the return value. The OCAP implementation shall grant them to the application.

The modified set of Permissions shall be a subset of the requested set of Permissions specified by the permissionInfo parameter, and shall be a superset of the set of the Permissions granted to unsigned applications (as returned by PermissionInformation.getUnsignedAppPermissions()).

**Parameters:**
`permissionInfo` - The PermissionInformation that specifies the application to be launched and its requested set of Permissions that are requested in a permission request file and requested for the unsigned application.
**Returns:**

An instance of a subclass of the java.security.PermissionCollection that contains a modified set of Permissions to be granted to the application specified by the permissionInfo parameter. The modified set of Permissions (i.e., return value) shall be granted to the application. If the modified set of Permissions is not a subset of the requested Permissions, or is not a superset of the set of the Permissions granted to unsigned applications (as returned by PermissionInformation.getUnsignedAppPermissions()), the OCAP implementation shall ignore the returned PermissionCollection and shall grant the requested set of Permissions to the application.

# Annex H OCAP 1.1 MPEG Component API

This annex defines the org.ocap.mpeg APIs.

***Table H–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| Annex H OCAP 1.1 MPEG Component API | No Corresponding Section | OCAP-Specific Extension |

## Package org.ocap.mpeg

| Class Summary | |
|---|---|
| **PODExtendedChannel** | This class represents an extended channel that provides access to private section data flows. |

### org.ocap.mpeg
### Class PODExtendedChannel

```
java.lang.Object
  └ org.davic.mpeg.TransportStream
      └ org.ocap.mpeg.PODExtendedChannel

public abstract class PODExtendedChannel
extends org.davic.mpeg.TransportStream
```

This class represents an extended channel that provides access to private section data flows. The extended channel is defined in the Host-POD Interface Standard (SCTE 28). When this class is specified as the stream parameter of the org.davic.mpeg.sections.SectionFilterGroup.attach(TransportStream, ResourceClient, Object) method, the SectionFilterGroup is connected to the extended channel, i.e., the filters in the SectionFilterGroup filter the private section data via OOB. The extended channel flow to be opened is specified by PID, when the org.davic.mpeg.sections.SectionFilter.startFiltering() method is called.

The methods defined in the super class (org.davic.mpeg.TransportStream) shall behave as follows:

- The getTransportStreamId() method returns -1.
- The retrieveService(int serviceId) method returns null.
- The retrieveServices() method returns null.

| Constructor Summary | |
|---|---|
| protected | **PODExtendedChannel**()<br>OCAP applications SHALL NOT use this method - it is provided for internal use by the OCAP implementation. |

# Method Summary

| | |
|---|---|
| `static PODExtendedChannel` | **`getInstance`**`()`<br>Gets a PODExtendedChannel instance. |

| **Methods inherited from class org.davic.mpeg.TransportStream** |
|---|
| `getTransportStreamId, retrieveService, retrieveServices` |

| **Methods inherited from class java.lang.Object** |
|---|
| `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait` |

# Constructor Detail

### PODExtendedChannel

`protected` **`PODExtendedChannel`**`()`

OCAP applications SHALL NOT use this method - it is provided for internal use by the OCAP implementation. The result of calling this method from an application is undefined, and valid implementations MAY throw any Error or RuntimeException.

# Method Detail

### getInstance

`public static PODExtendedChannel` **`getInstance`**`()`

Gets a PODExtendedChannel instance. The implementation MAY return the same instance each time, or it MAY return different (but functionally identical) instances.

**Returns:**

A PODExtendedChannel instance.

# Annex I OCAP 1.1 Net API

***Table I–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| Annex I OCAP 1.1 Net API | No Corresponding Section | OCAP-Specific Extension |

## Package org.ocap.net

| Class Summary | |
|---|---|
| **OcapLocator** | This class encapsulates an OCAP URL into an object. |
| **OCRCInterface** | This class models a return channel interface for use in receiving and transmitting IP packets over an OCAP-compliant return channel. |

### org.ocap.net
### Class OcapLocator

```
java.lang.Object
  └ org.davic.net.Locator
      └ org.ocap.net.OcapLocator
```

**All Implemented Interfaces:**
javax.tv.locator.Locator

```
public class OcapLocator
extends org.davic.net.Locator
```

This class encapsulates an OCAP URL into an object. This class provides access to locations of various types of items in the transport stream.

Note that the org.davic.net.Locator (super class of the OcapLocator) is modified in Section 12.2.1.9.4 *Content Referencing* of this specification as following:

org.davic.net.Locator implements javax.tv.locator.Locator
The javax.tv.locator.Locator.toExternalForm() method returns an OCAP URL string that is used to create an OcapLocator instance, in canonical form. If an OCAP locator is in canonical form, the following MUST hold:

- no character is escaped if it is possible to represent it without escaping according to the OCAP URL BNF. (E.g. "%41" is changed to "A").
- hex numbers do not have leading zeros, except for the number zero itself, which is represented as "0x0". (E.g. "0x01" is changed to "0x1").
- all instances of ISO_639_language_code must be lowercase. (E.g. "SPA" is changed to "spa").

No other change is performed to convert an OCAP locator to its canonical form.

All methods defined in this class that return Strings, except for toExternalForm(), return the String in Unicode format. I.e. They MUST un-escape characters in the corresponding portion of the URL that are escaped with the %nn syntax (where that syntax is permitted by the OCAP URL BNF), and they MUST UTF-8 decode the string.

All constructors defined in this class that take String parameters, except for the OcapLocator(String url) constructor, require the String in Unicode format. I.e. Where permitted by the OCAP URL BNF they MUST UTF-8 encode the string and they MUST escape (using the %nn syntax) any characters that require escaping. They MUST NOT escape any character that can be represented without escaping.

**See Also:**

`Locator`, `AppAttributes.getServiceLocator()`

# Constructor Summary

| |
|---|
| **`OcapLocator`**`(int sourceID)`<br>A constructor of this class corresponding to the OCAP URL form "ocap://source_id". |
| **`OcapLocator`**`(int frequency, int modulationFormat)`<br>A constructor of this class corresponding to the OCAP URL form "ocap://f=frequency[.m=modulation_format]". |
| **`OcapLocator`**`(int sourceID, int[] PID, int eventID, java.lang.String pathSegments)`<br>A constructor of this class corresponding to the OCAP URL form "ocap://source_id[. |
| **`OcapLocator`**`(int frequency, int programNumber, int modulationFormat)`<br>A constructor of this class corresponding to the OCAP URL form "ocap://(oobfdc\|f=frequency).program_number[.m=modulation_format]". |
| **`OcapLocator`**`(int sourceID, int eventID, int[] componentTags, java.lang.String pathSegments)`<br>A constructor of this class corresponding to the OCAP URL form "ocap://source_id[. |
| **`OcapLocator`**`(int frequency, int programNumber, int modulationFormat, int[] PID, int eventID, java.lang.String pathSegments)`<br>A constructor of this class corresponding to the OCAP URL form "ocap://(oobfdc\|f=frequency).program_number[.m=modulation_format] [. |
| **`OcapLocator`**`(int frequency, int programNumber, int modulationFormat, int eventID, int[] componentTags, java.lang.String pathSegments)`<br>A constructor of this class corresponding to the OCAP URL form "ocap://(oobfdc\|f=frequency).program_number[.m=modulation_format] [. |
| **`OcapLocator`**`(int frequency, int programNumber, int modulationFormat, short[] streamType, int[] index, int eventID, java.lang.String pathSegments)`<br>A constructor of this class corresponding to the OCAP URL form "ocap://(oobfdc\|f=frequency).program_number. |
| **`OcapLocator`**`(int frequency, int programNumber, int modulationFormat, short[] streamType, java.lang.String[] ISO639LanguageCode, int eventID, java.lang.String pathSegments)`<br>A constructor of this class corresponding to the OCAP URL form "ocap://(oobfdc\|f=frequency).program_number[.m=modulation_format] [.stream_type[,ISO_639_language_code] {&stream_type[,ISO_639_language_code]}][;event_id]{/path_segments}". |
| **`OcapLocator`**`(int frequency, int programNumber, int modulationFormat, java.lang.String[] componentName, int eventID, java.lang.String pathSegments)`<br>A constructor of this class corresponding to the OCAP URL form "ocap://(oobfdc\|f=frequency).program_number[.m=modulation_format] [. |

## Constructor Summary

| |
|---|
| **OcapLocator**(int sourceID, short[] streamType, int[] index, int eventID, java.lang.String pathSegments)<br>A constructor of this class corresponding to the OCAP URL form "ocap://source_id[.stream_type[,index]{&stream_type[,index]}] [;event_id]{/path_segments}". |
| **OcapLocator**(int sourceID, short[] streamType, java.lang.String[] ISO639LanguageCode, int eventID, java.lang.String pathSegments)<br>A constructor of this class corresponding to the OCAP URL form "ocap://source_id[.stream_type[,ISO_639_language_code]{& stream_type[,ISO_639_language_code]}][;event_id]{/path_segments}". |
| **OcapLocator**(int sourceID, java.lang.String[] componentName, int eventID, java.lang.String pathSegments)<br>A constructor of this class corresponding to the OCAP URL form "ocap://source_id[. |
| **OcapLocator**(java.lang.String url)<br>A constructor of this class for any form of OCAP URL. |
| **OcapLocator**(java.lang.String serviceName, int[] PID, int eventID, java.lang.String pathSegments)<br>A constructor of this class corresponding to the OCAP URL form "ocap://n=service_name[. |
| **OcapLocator**(java.lang.String serviceName, int eventID, int[] componentTags, java.lang.String pathSegments)<br>A constructor of this class corresponding to the OCAP URL form "ocap://n=service_name[. |
| **OcapLocator**(java.lang.String serviceName, short[] streamType, int[] index, int eventID, java.lang.String pathSegments)<br>A constructor of this class corresponding to the OCAP URL form "ocap://n=service_name[.stream_type[,index]{&stream_type[,index]}] [;event_id]{/path_segments}". |
| **OcapLocator**(java.lang.String serviceName, short[] streamType, java.lang.String[] ISO639LanguageCode, int eventID, java.lang.String pathSegments)<br>A constructor of this class corresponding to the OCAP URL form "ocap://n=service_name[.stream_type[,ISO_639_language_code]{& stream_type[,ISO_639_language_code]}][;event_id]{/path_segments}". |
| **OcapLocator**(java.lang.String serviceName, java.lang.String[] componentName, int eventID, java.lang.String pathSegments)<br>A constructor of this class corresponding to the OCAP URL form "ocap://n=service_name[. |

## Method Summary

| | |
|---|---|
| java.lang.String[] | **getComponentNames**()<br>This method returns a component_name value of the OCAP URL represented by this OcapLocator instance. |
| int[] | **getComponentTags**()<br>This method returns a component_tag value of the OCAP URL represented by this OcapLocator instance. |
| int | **getEventId**()<br>This method returns an event_id value of the OCAP URL represented by this OcapLocator instance. |

| Method Summary | |
|---|---|
| int | **getFrequency**()<br>This method returns a frequency value, in hertz, of the OCAP URL represented by this OcapLocator instance. |
| int[] | **getIndexes**()<br>This method returns an index value of the OCAP URL represented by this OcapLocator instance. |
| java.lang.String[] | **getLanguageCodes**()<br>This method returns an ISO_639_language_code value of the OCAP URL represented by this OcapLocator instance. |
| int | **getModulationFormat**()<br>This method returns a value representing a modulation_format as specified in SCTE 65. |
| java.lang.String | **getPathSegments**()<br>This method returns a path_segments string of the OCAP URL represented by this OcapLocator instance. |
| int[] | **getPIDs**()<br>This method returns a PID value of the OCAP URL represented by this OcapLocator instance. |
| int | **getProgramNumber**()<br>This method returns a program_number value of the OCAP URL represented by this OcapLocator instance. |
| java.lang.String | **getServiceName**()<br>This method returns a service_name value of the OCAP URL represented by this OcapLocator instance. |
| int | **getSourceID**()<br>This method returns a source_id value of the OCAP URL represented by this OcapLocator instance. |
| short[] | **getStreamTypes**()<br>This method returns a stream_type value of the OCAP URL represented by this OcapLocator instance. |

| Methods inherited from class org.davic.net.Locator |
|---|
| hasMultipleTransformations, toExternalForm, toString |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

| Methods inherited from interface javax.tv.locator.Locator |
|---|
| equals, hashCode |

# Constructor Detail

### OcapLocator

```
public OcapLocator(int sourceID)
            throws org.davic.net.InvalidLocatorException
```

A constructor of this class corresponding to the OCAP URL form "ocap://source_id".

Note that the OcapLocator does not automatically transform this OCAP URL BNF form to any other form, even if the getFrequency() etc. is called.

**Parameters:**
`sourceID` - a source_id value for the OCAP URL.
**Throws:**
`org.davic.net.InvalidLocatorException` - if the sourceID to construct the locator doesn't specify a valid OCAP URL (e.g. a value is out of range).

### OcapLocator

```
public OcapLocator(int frequency,
                   int programNumber,
                   int modulationFormat)
            throws org.davic.net.InvalidLocatorException
```

A constructor of this class corresponding to the OCAP URL form "ocap://(oobfdc|f=frequency).program_number[.m=modulation_format]".

Note that the OcapLocator does not automatically transform this OCAP URL BNF form to any other form, even if the getSourceId() etc. is called.

**Parameters:**
`frequency` - a frequency value for the OCAP URL in hertz. If the value is -1 then "oobfdc" is used instead of the frequency term and the modulationFormat parameter is ignored.
`programNumber` - a program_number value for the OCAP URL
`modulationFormat` - a value representing a modulation_format as specified in SCTE 65. If the value is -1 the modulation_format is not specified and the modulation_format term will not be included in the locator constructed.
**Throws:**
`org.davic.net.InvalidLocatorException` - if the parameters to construct the locator don't specify a valid OCAP URL (e.g. a value is out of range).

### OcapLocator

```
public OcapLocator(int frequency,
                   int modulationFormat)
            throws org.davic.net.InvalidLocatorException
```

A constructor of this class corresponding to the OCAP URL form "ocap://f=frequency[.m=modulation_format]".

Note that the OcapLocator does not automatically transform this OCAP URL BNF form to any other form, even if the getSourceId() etc. is called.

**Parameters:**
`frequency` - a frequency value for the OCAP URL in hertz.

`modulationFormat` - a value representing a modulation_format as specified in SCTE 65. If the value is -1 the modulation_format is not specified and the modulation_format term will not be included in the locator constructed.

**Throws:**

`org.davic.net.InvalidLocatorException` - if the parameters to construct the locator don't specify a valid OCAP URL (e.g. a value is out of range).

### OcapLocator

```
public OcapLocator(java.lang.String url)
            throws org.davic.net.InvalidLocatorException
```

A constructor of this class for any form of OCAP URL.

Note that the OcapLocator does not automatically transform the specified url string to any other form, even if any get methods for the value that is not included in the url string are called.

**Parameters:**

`url` - a string expression that represents the OCAP URL.

**Throws:**

`org.davic.net.InvalidLocatorException` - if the url to construct the locator doesn't specify a valid OCAP URL (e.g. a value is out of range).

### OcapLocator

```
public OcapLocator(int sourceID,
                   short[] streamType,
                   java.lang.String[] ISO639LanguageCode,
                   int eventID,
                   java.lang.String pathSegments)
            throws org.davic.net.InvalidLocatorException
```

A constructor of this class corresponding to the OCAP URL form "ocap://source_id[.stream_type[,ISO_639_language_code]{& stream_type[,ISO_639_language_code]}][;event_id]{/path_segments}". Some of the parameters can be omitted according to the OCAP URL BNF definition.

Note that the OcapLocator does not automatically transform this OCAP URL BNF form to any other form, even if the getFrequency() etc. is called.

**Parameters:**

`sourceID` - a source_id value for the OCAP URL.

`streamType` - a stream_type value for the OCAP URL. A combination of the streamType[n] and the ISO639LanguageCode[n] makes a program_element. The streamType shall be a zero length array, if it is omitted in the OCAP URL.

`ISO639LanguageCode` - an ISO_639_language_code value for the OCAP URL. A combination of the streamType[n] and the ISO639LanguageCode[n] makes a program_element. The ISO639LanguageCode shall be a zero length array, if it is omitted in the OCAP URL. If ISO639LanguageCode is not a zero-length array, it shall be an array with the same length as streamType. If ISO639LanguageCode[n] is null, then the language code for streamType[n] is omitted in the OCAP URL.

`eventID` - an event_id value for the OCAP URL. The event_id shall be -1, if it is omitted in the OCAP URL.

`pathSegments` - a path_segments value for the OCAP URL. The pathSegments shall be null, if it is omitted in the OCAP URL.

**Throws:**

`org.davic.net.InvalidLocatorException` - if the parameters to construct the locator don't specify a valid OCAP URL (e.g. a value is out of range).

### OcapLocator

```
public OcapLocator(int sourceID,
                   short[] streamType,
                   int[] index,
                   int eventID,
                   java.lang.String pathSegments)
            throws org.davic.net.InvalidLocatorException
```

A constructor of this class corresponding to the OCAP URL form "ocap://source_id[.stream_type[,index]{&stream_type[,index]}] [;event_id]{/path_segments}". Some of parameters can be omitted according to the OCAP URL BNF definition.

Note that the OcapLocator does not automatically transform this OCAP URL BNF form to any other form, even if the getFrequency() etc. is called.

**Parameters:**

`sourceID` - a source_id value for the OCAP URL.

`streamType` - a stream_type value for the OCAP URL. A combination of the streamType[n] and the index[n] makes a program_element. The streamType shall be a zero length array, if it is omitted in the OCAP URL.

`index` - an index value for the OCAP URL. A combination of the streamType[n] and the index[n] makes a program_element. The index shall be a zero length array, if it is omitted in the OCAP URL. If index is not a zero-length array, it shall be an array with the same length as streamType. If index[n] is -1, then the index for streamType[n] is omitted in the OCAP URL.

`eventID` - an event_id value for the OCAP URL. The event_id shall be -1, if it is omitted in the OCAP URL.

`pathSegments` - a path_segments value for the OCAP URL. The pathSegments shall be null, if it is omitted in the OCAP URL.

**Throws:**

`org.davic.net.InvalidLocatorException` - if the parameters to construct the locator don't specify a valid OCAP URL (e.g. a value is out of range).

### OcapLocator

```
public OcapLocator(int sourceID,
                   int[] PID,
                   int eventID,
                   java.lang.String pathSegments)
            throws org.davic.net.InvalidLocatorException
```

A constructor of this class corresponding to the OCAP URL form "ocap://source_id[.+PID{&PID}][;event_id]{/path_segments}". Some of parameters can be omitted according to the OCAP URL BNF definition.

Note that the OcapLocator does not automatically transform this OCAP URL BNF form to any other form, even if the getFrequency() etc. is called.

**Parameters:**

`sourceID` - a source_id value for the OCAP URL.

`PID` - a PID value for the OCAP URL. The PID shall be a zero length array, if it is omitted in the OCAP URL.

`eventID` - an event_id value for the OCAP URL. The event_id shall be -1, if it is omitted in the OCAP URL.
`pathSegments` - a path_segments value for the OCAP URL. The pathSegments shall be null, if it is omitted in the OCAP URL.
**Throws:**
`org.davic.net.InvalidLocatorException` - if the parameters to construct the locator don't specify a valid OCAP URL (e.g. a value is out of range).

### OcapLocator

```
public OcapLocator(int sourceID,
                   java.lang.String[] componentName,
                   int eventID,
                   java.lang.String pathSegments)
            throws org.davic.net.InvalidLocatorException
```

A constructor of this class corresponding to the OCAP URL form "ocap://source_id[.$component_name{&component_name}] [;event_id]{/path_segments}". Some of parameters can be omitted according to the OCAP URL BNF definition.

Note that the OcapLocator does not automatically transform this OCAP URL BNF form to any other form, even if the getFrequency() etc. is called.

**Parameters:**
`sourceID` - a source_id value for the OCAP URL.
`componentName` - a component_name value for the OCAP URL. The component_name shall be a zero length array, if it is omitted in the OCAP URL.
`eventID` - an event_id value for the OCAP URL. The event_id shall be -1, if it is omitted in the OCAP URL.
`pathSegments` - a path_segments value for the OCAP URL. The pathSegments shall be null, if it is omitted in the OCAP URL.
**Throws:**
`org.davic.net.InvalidLocatorException` - if the parameters to construct the locator don't specify a valid OCAP URL (e.g. a value is out of range).

### OcapLocator

```
public OcapLocator(int sourceID,
                   int eventID,
                   int[] componentTags,
                   java.lang.String pathSegments)
            throws org.davic.net.InvalidLocatorException
```

A constructor of this class corresponding to the OCAP URL form "ocap://source_id[.@component_tag{&component_tag}][;event_id]{/path_segments}". Some of parameters can be omitted according to the OCAP URL BNF definition.

Note that the OcapLocator does not automatically transform this OCAP URL BNF form to any other form, even if the getFrequency() etc. is called.

**WARNING:** Note that the parameter order for this constructor is different from other OcapLocator constructors - the eventId is *before* the componentTags. If you are an OCAP application author and you get it wrong, your program will compile and run but it will be calling the constructor that expects a list of PIDs instead.

**Parameters:**
`sourceID` - a source_id value for the OCAP URL.

`eventID` - an event_id value for the OCAP URL. The event_id shall be -1, if it is omitted in the OCAP URL.
`componentTags` - a component_tag value for the OCAP URL. The component_tag shall be a zero length array, if it is omitted in the OCAP URL.
`pathSegments` - a path_segments value for the OCAP URL. The pathSegments shall be null, if it is omitted in the OCAP URL.
**Throws:**
`org.davic.net.InvalidLocatorException` - if the parameters to construct the locator don't specify a valid OCAP URL (e.g. a value is out of range).

### OcapLocator

```
public OcapLocator(java.lang.String serviceName,
                   short[] streamType,
                   java.lang.String[] ISO639LanguageCode,
                   int eventID,
                   java.lang.String pathSegments)
            throws org.davic.net.InvalidLocatorException
```

A constructor of this class corresponding to the OCAP URL form "ocap://n=service_name[.stream_type[,ISO_639_language_code]{& stream_type[,ISO_639_language_code]}][;event_id]{/path_segments}". Some of parameters can be omitted according to the OCAP URL BNF definition.

Note that the OcapLocator does not automatically transform this OCAP URL BNF form to any other form, even if the getFrequency() etc. is called.

**Parameters:**
`serviceName` - a service_name value for the OCAP URL.
`streamType` - a stream_type value for the OCAP URL. A combination of the streamType[n] and the ISO639LanguageCode[n] makes a program_element. The streamType shall be a zero length array, if it is omitted in the OCAP URL.
`ISO639LanguageCode` - an ISO_639_language_code value for the OCAP URL. A combination of the streamType[n] and the ISO639LanguageCode[n] makes a program_element. The ISO639LanguageCode shall be a zero length array, if it is omitted in the OCAP URL.
`eventID` - an event_id value for the OCAP URL. The event_id shall be -1, if it is omitted in the OCAP URL.
`pathSegments` - a path_segments value for the OCAP URL. The pathSegments shall be null, if it is omitted in the OCAP URL.
**Throws:**
`org.davic.net.InvalidLocatorException` - if the parameters to construct the locator don't specify a valid OCAP URL (e.g. a value is out of range).

### OcapLocator

```
public OcapLocator(java.lang.String serviceName,
                   short[] streamType,
                   int[] index,
                   int eventID,
                   java.lang.String pathSegments)
            throws org.davic.net.InvalidLocatorException
```

A constructor of this class corresponding to the OCAP URL form "ocap://n=service_name[.stream_type[,index]{&stream_type[,index]}] [;event_id]{/path_segments}". Some of parameters can be omitted according to the OCAP URL BNF definition.

Note that the OcapLocator does not automatically transform this OCAP URL BNF form to any other form, even if the getFrequency() etc. is called.

**Parameters:**
`serviceName` - a service_name value for the OCAP URL.
`streamType` - a stream_type value for the OCAP URL. A combination of the streamType[n] and the index[n] makes a program_element. The streamType shall be a zero length array, if it is omitted in the OCAP URL.
`index` - an index value for the OCAP URL. A combination of the streamType[n] and the index[n] makes a program_element. The index shall be a zero length array, if it is omitted in the OCAP URL.
`eventID` - an event_id value for the OCAP URL. The event_id shall be -1, if it is omitted in the OCAP URL.
`pathSegments` - a path_segments value for the OCAP URL. The pathSegments shall be null, if it is omitted in the OCAP URL.
**Throws:**
`org.davic.net.InvalidLocatorException` - if the parameters to construct the locator don't specify a valid OCAP URL (e.g. a value is out of range).

### OcapLocator

```
public OcapLocator(java.lang.String serviceName,
                   int[] PID,
                   int eventID,
                   java.lang.String pathSegments)
            throws org.davic.net.InvalidLocatorException
```

A constructor of this class corresponding to the OCAP URL form "ocap://n=service_name[.+PID{&PID}][;event_id]{/path_segments}". Some of parameters can be omitted according to the OCAP URL BNF definition.

Note that the OcapLocator does not automatically transform this OCAP URL BNF form to any other form, even if the getFrequency() etc. is called.

**Parameters:**
`serviceName` - a service_name value for the OCAP URL.
`PID` - a PID value for the OCAP URL. The PID shall be a zero length array, if it is omitted in the OCAP URL.
`eventID` - an event_id value for the OCAP URL. The event_id shall be -1, if it is omitted in the OCAP URL.
`pathSegments` - a path_segments value for the OCAP URL. The pathSegments shall be null, if it is omitted in the OCAP URL.
**Throws:**
`org.davic.net.InvalidLocatorException` - if the parameters to construct the locator don't specify a valid OCAP URL (e.g. a value is out of range).

### OcapLocator

```
public OcapLocator(java.lang.String serviceName,
                   java.lang.String[] componentName,
                   int eventID,
                   java.lang.String pathSegments)
            throws org.davic.net.InvalidLocatorException
```

A constructor of this class corresponding to the OCAP URL form "ocap://n=service_name[.$component_name{&component_name}] [;event_id]{/path_segments}". Some of parameters can be omitted according to the OCAP URL BNF definition.

Note that the OcapLocator does not automatically transform this OCAP URL BNF form to any other form, even if the getFrequency() etc. is called.

**Parameters:**
`serviceName` - a service_name value for the OCAP URL.
`componentName` - a component_name value for the OCAP URL. The component_name shall be a zero length array, if it is omitted in the OCAP URL.
`eventID` - an event_id value for the OCAP URL. The event_id shall be -1, if it is omitted in the OCAP URL.
`pathSegments` - a path_segments value for the OCAP URL. The pathSegments shall be null, if it is omitted in the OCAP URL.
**Throws:**
`org.davic.net.InvalidLocatorException` - if the parameters to construct the locator don't specify a valid OCAP URL (e.g. a value is out of range).

### OcapLocator

```
public OcapLocator(java.lang.String serviceName,
                   int eventID,
                   int[] componentTags,
                   java.lang.String pathSegments)
            throws org.davic.net.InvalidLocatorException
```

A constructor of this class corresponding to the OCAP URL form "ocap://n=service_name[.@component_tag{&component_tag}][;event_id]{/path_segments}". Some of parameters can be omitted according to the OCAP URL BNF definition.

Note that the OcapLocator does not automatically transform this OCAP URL BNF form to any other form, even if the getFrequency() etc. is called.

**WARNING:** Note that the parameter order for this constructor is different from other OcapLocator constructors - the eventId is *before* the componentTags. If you are an OCAP application author and you get it wrong, your program will compile and run but it will be calling the constructor that expects a list of PIDs instead.

**Parameters:**
`serviceName` - a service_name value for the OCAP URL.
`eventID` - an event_id value for the OCAP URL. The event_id shall be -1, if it is omitted in the OCAP URL.
`componentTags` - a component_tag value for the OCAP URL. The component_tag shall be a zero length array, if it is omitted in the OCAP URL.
`pathSegments` - a path_segments value for the OCAP URL. The pathSegments shall be null, if it is omitted in the OCAP URL.
**Throws:**
`org.davic.net.InvalidLocatorException` - if the parameters to construct the locator don't specify a valid OCAP URL (e.g. a value is out of range).

### OcapLocator

```
public OcapLocator(int frequency,
                   int programNumber,
                   int modulationFormat,
                   short[] streamType,
                   java.lang.String[] ISO639LanguageCode,
                   int eventID,
                   java.lang.String pathSegments)
            throws org.davic.net.InvalidLocatorException
```

A constructor of this class corresponding to the OCAP URL form "ocap://(oobfdc|f=frequency).program_number[.m=modulation_format] [.stream_type[,ISO_639_language_code] {&stream_type[,ISO_639_language_code]}][;event_id]{/path_segments}". Some of parameters can be omitted according to the OCAP URL BNF definition.

Note that the OcapLocator does not automatically transform this OCAP URL BNF form to any other form, even if the getFrequency() etc. is called.

**Parameters:**
`frequency` - a frequency value for the OCAP URL in hertz. If the value is -1 then "oobfdc" is used instead of the frequency term and the modulationFormat parameter is ignored.
`programNumber` - a program_number value for the OCAP URL
`modulationFormat` - a value representing a modulation_format as specified in SCTE 65. If the value is -1 the modulation_format is not specified and the modulation_format term will not be included in the locator constructed.
`streamType` - a stream_type value for the OCAP URL. A combination of the streamType[n] and the ISO639LanguageCode[n] makes a program_element. The streamType shall be a zero length array, if it is omitted in the OCAP URL.
`ISO639LanguageCode` - an ISO_639_language_code value for the OCAP URL. A combination of the streamType[n] and the ISO639LanguageCode[n] makes a program_element. The ISO639LanguageCode shall be a zero length array, if it is omitted in the OCAP URL. If ISO639LanguageCode is not a zero-length array, it shall be an array with the same length as streamType. If ISO639LanguageCode[n] is null, then the language code for streamType[n] is omitted in the OCAP URL.
`eventID` - an event_id value for the OCAP URL. The event_id shall be -1, if it is omitted in the OCAP URL.
`pathSegments` - a path_segments value for the OCAP URL. The pathSegments shall be null, if it is omitted in the OCAP URL.
**Throws:**
`org.davic.net.InvalidLocatorException` - if the parameters to construct the locator don't specify a valid OCAP URL (e.g. a value is out of range).

### OcapLocator

```
public OcapLocator(int frequency,
                   int programNumber,
                   int modulationFormat,
                   short[] streamType,
                   int[] index,
                   int eventID,
                   java.lang.String pathSegments)
            throws org.davic.net.InvalidLocatorException
```

A constructor of this class corresponding to the OCAP URL form "ocap://(oobfdc|f=frequency).program_number.[m=modulation_format] [.stream_type[,index]{&stream_type[,index]}] [;event_id]{/path_segments}". Some of parameters can be omitted according to the OCAP URL BNF definition.

Note that the OcapLocator does not automatically transform this OCAP URL BNF form to any other form, even if the getFrequency() etc. is called.

**Parameters:**
`frequency` - a frequency value for the OCAP URL in hertz. If the value is -1 then "oobfdc" is used instead of the frequency term and the modulationFormat parameter is ignored.
`programNumber` - a program_number value for the OCAP URL

`modulationFormat` - a value representing a modulation_format as specified in SCTE 65. If the value is -1 the modulation_format is not specified and the modulation_format term will not be included in the locator constructed.
`streamType` - a stream_type value for the OCAP URL. A combination of the streamType[n] and the index[n] makes a program_element. The streamType shall be a zero length array, if it is omitted in the OCAP URL.
`index` - an index value for the OCAP URL. A combination of the streamType[n] and the index[n] makes a program_element. The index shall be a zero length array, if it is omitted in the OCAP URL. If index is not a zero-length array, it shall be an array with the same length as streamType. If index[n] is -1, then the index for streamType[n] is omitted in the OCAP URL.
`eventID` - an event_id value for the OCAP URL. The event_id shall be -1, if it is omitted in the OCAP URL.
`pathSegments` - a path_segments value for the OCAP URL. The pathSegments shall be null, if it is omitted in the OCAP URL.
**Throws:**
`org.davic.net.InvalidLocatorException` - if the parameters to construct the locator don't specify a valid OCAP URL (e.g. a value is out of range).

### OcapLocator

```
public OcapLocator(int frequency,
                   int programNumber,
                   int modulationFormat,
                   int[] PID,
                   int eventID,
                   java.lang.String pathSegments)
            throws org.davic.net.InvalidLocatorException
```

A constructor of this class corresponding to the OCAP URL form "ocap://(oobfdc|f=frequency).program_number[.m=modulation_format] [.+PID{&PID}][;event_id]{/path_segments}". Some of parameters can be omitted according to the OCAP URL BNF definition.

Note that the OcapLocator does not automatically transform this OCAP URL BNF form to any other form, even if the getFrequency() etc. is called.

**Parameters:**
`frequency` - a frequency value for the OCAP URL in hertz. If the value is -1 then "oobfdc" is used instead of the frequency term and the modulationFormat parameter is ignored.
`programNumber` - a program_number value for the OCAP URL
`modulationFormat` - a value representing a modulation_format as specified in SCTE 65. If the value is -1 the modulation_format is not specified and the modulation_format term will not be included in the locator constructed.
`PID` - a PID value for the OCAP URL. The PID shall be a zero length array, if it is omitted in the OCAP URL.
`eventID` - an event_id value for the OCAP URL. The event_id shall be -1, if it is omitted in the OCAP URL.
`pathSegments` - a path_segments value for the OCAP URL. The pathSegments shall be null, if it is omitted in the OCAP URL.
**Throws:**
`org.davic.net.InvalidLocatorException` - if the parameters to construct the locator don't specify a valid OCAP URL (e.g. a value is out of range).

### OcapLocator

```
public OcapLocator(int frequency,
                   int programNumber,
                   int modulationFormat,
                   java.lang.String[] componentName,
                   int eventID,
                   java.lang.String pathSegments)
            throws org.davic.net.InvalidLocatorException
```

A constructor of this class corresponding to the OCAP URL form "ocap://(oobfdc|f=frequency).program_number[.m=modulation_format] [.$component_name{&component_name}] [;event_id]{/path_segments}". Some of parameters can be omitted according to the OCAP URL BNF definition.

Note that the OcapLocator does not automatically transform this OCAP URL BNF form to any other form, even if the getFrequency() etc. is called.

**Parameters:**

`frequency` - a frequency value for the OCAP URL in hertz. If the value is -1 then "oobfdc" is used instead of the frequency term and the modulationFormat parameter is ignored.

`programNumber` - a program_number value for the OCAP URL

`modulationFormat` - a value representing a modulation_format as specified in SCTE 65. If the value is -1 the modulation_format is not specified and the modulation_format term will not be included in the locator constructed.

`componentName` - a component_name value for the OCAP URL. The component_name shall be a zero length array, if it is omitted in the OCAP URL.

`eventID` - an event_id value for the OCAP URL. The event_id shall be -1, if it is omitted in the OCAP URL.

`pathSegments` - a path_segments value for the OCAP URL. The pathSegments shall be null, if it is omitted in the OCAP URL.

**Throws:**

`org.davic.net.InvalidLocatorException` - if the parameters to construct the locator don't specify a valid OCAP URL (e.g. a value is out of range).

### OcapLocator

```
public OcapLocator(int frequency,
                   int programNumber,
                   int modulationFormat,
                   int eventID,
                   int[] componentTags,
                   java.lang.String pathSegments)
            throws org.davic.net.InvalidLocatorException
```

A constructor of this class corresponding to the OCAP URL form "ocap://(oobfdc|f=frequency).program_number[.m=modulation_format] [.@component_tag{&component_tag}][;event_id]{/path_segments}". Some of parameters can be omitted according to the OCAP URL BNF definition.

Note that the OcapLocator does not automatically transform this OCAP URL BNF form to any other form, even if the getFrequency() etc. is called.

**WARNING:** Note that the parameter order for this constructor is different from other OcapLocator constructors - the eventId is *before* the componentTags. If you are an OCAP application author and you get it wrong, your program will compile and run but it will be calling the constructor that expects a list of PIDs instead.

**Parameters:**

`frequency` - a frequency value for the OCAP URL in hertz. If the value is -1 then "oobfdc" is used instead of the frequency term and the modulationFormat parameter is ignored.
`programNumber` - a program_number value for the OCAP URL
`modulationFormat` - a value representing a modulation_format as specified in SCTE 65. If the value is -1 the modulation_format is not specified and the modulation_format term will not be included in the locator constructed.
`eventID` - an event_id value for the OCAP URL. The event_id shall be -1, if it is omitted in the OCAP URL.
`componentTags` - a component_tag value for the OCAP URL. The component_tag shall be a zero length array, if it is omitted in the OCAP URL.
`pathSegments` - a path_segments value for the OCAP URL. The pathSegments shall be null, if it is omitted in the OCAP URL.
**Throws:**
`org.davic.net.InvalidLocatorException` - if the parameters to construct the locator don't specify a valid OCAP URL (e.g. a value is out of range).

# Method Detail

### getSourceID

`public int` **`getSourceID`**`()`

This method returns a source_id value of the OCAP URL represented by this OcapLocator instance.
**Returns:**
a source_id value of the OCAP URL represented by this OcapLocator instance. If the OCAP URL that is specified to construct an OcapLocator instance doesn't include it, -1 returns.

### getServiceName

`public java.lang.String` **`getServiceName`**`()`

This method returns a service_name value of the OCAP URL represented by this OcapLocator instance.
**Returns:**
a service_name value of the OCAP URL represented by this OcapLocator instance. If the OCAP URL that is specified to construct an OcapLocator instance doesn't include it, null returns.

### getFrequency

`public int` **`getFrequency`**`()`

This method returns a frequency value, in hertz, of the OCAP URL represented by this OcapLocator instance.
**Returns:**
a frequency value, in hertz, of the OCAP URL represented by this OcapLocator instance. If the OCAP URL that is specified to construct an OcapLocator instance doesn't include it or the locator is OOB, -1 is returned. If the getProgramNumber method returns a value other than -1, the locator is OOB.

### getModulationFormat

`public int` **`getModulationFormat`**`()`

This method returns a value representing a modulation_format as specified in SCTE 65.
**Returns:**

a value representing the modulation format. If the OCAP URL that is specified to construct an OcapLocator instance doesn't include it or -1 was passed in as the modulation format, -1 is returned. When the locator contains a frequency term and this method returns a -1, a default modulation format value of QAM256 is implied.

#### getProgramNumber

`public int` **`getProgramNumber`**`()`

This method returns a program_number value of the OCAP URL represented by this OcapLocator instance.

**Returns:**

a program_number value of the OCAP URL represented by this OcapLocator instance. If the OCAP URL that is specified to construct an OcapLocator instance doesn't include it, -1 returns.

#### getStreamTypes

`public short[]` **`getStreamTypes`**`()`

This method returns a stream_type value of the OCAP URL represented by this OcapLocator instance.

**Returns:**

a stream_type value of the OCAP URL represented by this OcapLocator instance. The order of stream_types is same as specified in the constructor. If the OCAP URL that is specified to construct an OcapLocator instance doesn't include it, a zero length array returns.

#### getLanguageCodes

`public java.lang.String[]` **`getLanguageCodes`**`()`

This method returns an ISO_639_language_code value of the OCAP URL represented by this OcapLocator instance.

**Returns:**

an ISO_639_language_code value of the OCAP URL represented by this OcapLocator instance. The order of ISO_639_language_code is same as specified in the constructor. If the OCAP URL that is specified to construct an OcapLocator instance doesn't include any language codes, a zero length array returns. If the OCAP URL that is specified to construct an OcapLocator instance includes any language codes, an array is returned that is the same length as that returned by getStreamTypes(). Some of the elements in this array may be null, if no language was specified for the corresponding stream_type.

#### getIndexes

`public int[]` **`getIndexes`**`()`

This method returns an index value of the OCAP URL represented by this OcapLocator instance.

**Returns:**

an index value of the OCAP URL represented by this OcapLocator instance. The order of index is same as specified in the constructor. If the OCAP URL that is specified to construct an OcapLocator instance doesn't include any indexes, a zero length array returns. If the OCAP URL that is specified to construct an OcapLocator instance includes any indexes, an array is returned that is the same length as that returned by getStreamTypes(). Some of the elements in this array may be -1, if no index was specified for the corresponding stream_type.

#### getEventId

`public int` **`getEventId`**`()`

This method returns an event_id value of the OCAP URL represented by this OcapLocator instance.

**Returns:**

an event_id value of the OCAP URL represented by this OcapLocator instance. If the OCAP URL that is specified to construct an OcapLocator instance doesn't include it, -1 returns.

#### getPIDs

`public int[]` **`getPIDs`**`()`

This method returns a PID value of the OCAP URL represented by this OcapLocator instance.

**Returns:**

a PID value of the OCAP URL represented by this OcapLocator instance. The order of PID is same as specified in the constructor. If the OCAP URL that is specified to construct an OcapLocator instance doesn't include it, a zero length array returns.

#### getComponentNames

`public java.lang.String[]` **`getComponentNames`**`()`

This method returns a component_name value of the OCAP URL represented by this OcapLocator instance.

**Returns:**

a component_name value of the OCAP URL represented by this OcapLocator instance. The order of component_name is same as specified in the constructor. If the OCAP URL that is specified to construct an OcapLocator instance doesn't include it, a zero length array returns.

#### getComponentTags

`public int[]` **`getComponentTags`**`()`

This method returns a component_tag value of the OCAP URL represented by this OcapLocator instance.

**Returns:**

a component_tag value of the OCAP URL represented by this OcapLocator instance. The order of component_tags is same as specified in the constructor. If the OCAP URL that is specified to construct an OcapLocator instance doesn't include it, a zero length array returns.

#### getPathSegments

`public java.lang.String` **`getPathSegments`**`()`

This method returns a path_segments string of the OCAP URL represented by this OcapLocator instance.

**Returns:**

a path_segments string of the OCAP URL represented by this OcapLocator instance. If the OCAP URL that is specified to construct an OcapLocator instance doesn't include it, null returns.

**org.ocap.net**
## Class OCRCInterface

```
java.lang.Object
  └ org.dvb.net.rc.RCInterface
      └ org.ocap.net.OCRCInterface
```

```
public class OCRCInterface
extends org.dvb.net.rc.RCInterface
```

This class models a return channel interface for use in receiving and transmitting IP packets over an OCAP-compliant return channel. This class does not model any concept of connection. Hence interfaces represented by this class are permanently connected. The getType() method inherited from org.dvb.net.rc.RCInterface SHALL return TYPE_CATV when called on an instance of this class.

## Field Summary

| | |
|---|---|
| static int | **SUBTYPE_CATV_DOCSIS**<br>Constant to indicate a DOCSIS return channel. |
| static int | **SUBTYPE_CATV_OOB**<br>Constant to indicate an OOB return channel, either SCTE DVS 167 or SCTE DVS 178, access through the CableCARD interface as specified in SCTE DVS 216. |

| **Fields inherited from class org.dvb.net.rc.RCInterface** |
|---|
| TYPE_CATV, TYPE_DECT, TYPE_ISDN, TYPE_LMDS, TYPE_MATV, TYPE_OTHER, TYPE_PSTN, TYPE_RCS, TYPE_UNKNOWN |

## Constructor Summary

| | |
|---|---|
| protected | **OCRCInterface**() |

## Method Summary

| | |
|---|---|
| int | **getSubType**()<br>Return the type of cable return channel. |

| **Methods inherited from class org.dvb.net.rc.RCInterface** |
|---|
| getDataRate, getType |

| Methods inherited from class java.lang.Object |
| --- |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Field Detail

### SUBTYPE_CATV_DOCSIS

```
public static final int SUBTYPE_CATV_DOCSIS
```

Constant to indicate a DOCSIS return channel.

### SUBTYPE_CATV_OOB

```
public static final int SUBTYPE_CATV_OOB
```

Constant to indicate an OOB return channel, either SCTE DVS 167 or SCTE DVS 178, access through the CableCARD interface as specified in SCTE DVS 216.

# Constructor Detail

### OCRCInterface

```
protected OCRCInterface()
```

# Method Detail

### getSubType

```
public int getSubType()
```

Return the type of cable return channel.

**Returns:**

the type of return channel represented by this object encoded as one of the constants (SUBTYPE_CATV_DOCSIS or SUBTYPE_CATV_OOB) defined in this class.

# Annex J Datagram Socket Buffer Control

This API is an extension to allow an application to manage the receive buffer size of a Java datagram.

## J.1 DVB-GEM and DVB-MHP Specification Correspondence

This section of the OCAP 1.1 Profile corresponds to Annex Q of [DVB-MHP1.1.2] and [DVB-GEM 1.1] as follows:

***Table J–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| Annex J Datagram Socket Buffer Control | Annex Q (normative): Datagram socket buffer control | Extension | Annex Q (normative): Datagram Socket Buffer Control | Extension |
| Annex J.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| Annex J.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| Annex J.2.1 Extensions to DVB/MHP (Normative) | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |
| Annex J.2.1.1 Package javax.tv.net Description | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | |

The OpenCable Application Platform is in complete compliance with the API, org.dvb.net, specified in this section.

## J.2 OCAP 1.1 Specific Requirements

### J.2.1 Extensions to DVB/MHP (Normative)

This information extends the specification requirements made to the [DVB-MHP1.1.2] and [DVB-GEM 1.1].

The following information is normative for the OCAP 1.1 Profile. It is specific to OCAP 1.1 and is not contained within the [DVB-MHP1.1.2] or [DVB-GEM 1.1].

#### *J.2.1.1 Package javax.tv.net Description*

This specification does not define how the system supports reception of broadcast IP data as defined by javax.tv.net.InterfaceMap.getLocalAddress().

The only form of communication is bi-direction IP traffic is using the out-of-band return channel for transmission of IP datagrams.

# Annex K OCAP 1.1 Event API

This section describes the OCAP User Input Event model. User Input Events are generated by the platform to indicate that the user has pressed a key on a remote control or keyboard. This API provides an application with two fundamentally different ways to catch and process these events. These are:

- through the standard `java.awt` event model
- through the `org.dvb.event` model

The standard `java.awt` model provides applications with a way to tie user input events to a specific component on the display. In order to use this API, the application SHALL have a visible AWT or HAVi component on the screen and that component SHALL have "focus" as defined by the AWT. In OCAP, applications can use the org.ocap.ui.event package that extends the `org.dvb.event` package and defines keycodes beyond those defined by java or HAVi.

The `org.dvb.event` model allows applications to receive notification of user input events even if they have no visible components being displayed at the time. While it is theoretically possible for an application to make use of both APIs, it will typically use only one of the two models.

The event object itself conveys the same information in either case; the type of event (Key Pressed/Typed/Released), the keycode, a time stamp, and an indicator of whether "special" key modifiers were also pressed (Ctrl, Alt, Shift, etc.). AWT events have an additional field which identifies the component which currently has focus. This is used so that a central event dispatcher within an application can determine how to internally process the event. The `org.dvb.event` API further extends the `java.awt` model by defining a mechanism for allowing applications to "exclusively reserve" an event keycode. If an event is exclusively reserved by an application, only that application will receive notification of the event. Otherwise, multiple applications may receive notification of the event occurrence. This capability is also built in to the `org.ocap.event` API as well.

The `org.ocap.event` model adds a key filter mechanism so that an application with permission, such as the Monitor Application, can receive an event before the requesting application(s) and perform the following operations:

Modify the event.
Route the event; that is, specify which of the requesting applications are to receive the event. The mechanism for specifying target applications is the UserEventAction object.

At most only one UserEventFilter at a time may be allowed to filter events. When requesting event filter capability from the Event Manager, the requesting application may define a key repository to enumerate the set of keys to be filtered. The Event Manager will maintain the order of key events; that is, the Event Manager will not process an event until all filtered event processing has been completed by the filtering UserEventFilter.

The org.ocap.event model adds a filtered repository mechanism so that an application with permission, such as the Monitor Application, can set a repository of key events that will be forwarded to the UserEventFilter. The filtered key repository cannot contain any of the mandatory ordinary key events as defined in Section 16.2.2.1.

A single application may register several listeners for a single event. The Event Manager is required to distribute events to all qualified listeners, regardless of whether a single application may receive multiple events. The set of applications specified by the UserEventFilter via the UserEventAction object SHALL maintain this behavior; that is, if multiple listeners are registered by an application that is listed in the UserEventAction object, then every qualified listener will receive the event.

## K.1 DVB-GEM and DVB-MHP Specification Correspondence

This annex of the OCAP 1.1 Profile corresponds to Annex J of [DVB-MHP1.1.2] and [DVB-GEM 1.1] as follows:

***Table K–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| Annex K OCAP 1.1 Event API | Annex J (normative): DVB-J event API | Extension | Annex J (normative): DVB-J event API | Extension |
| Annex K.1 DVB-GEM and DVB-MHP Specification Correspondence | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | OCAP-Specific Extension |
| Annex K.2 OCAP 1.1 Specific Requirements | No Corresponding Section | OCAP-Specific Extension | No Corresponding Section | OCAP-Specific Extension |

## K.2 OCAP 1.1 Specific Requirements

This section extends Annex J of [DVB-GEM 1.1]. The OCAP package, `org.ocap.event` functionally replaces the DVB-J package, `org.dvb.event and has the following extensions:`

- org.ocap.event.EventManager extends org.dvb.event.EventManager
- org.ocap.event.UserEvent extends org.dvb.event.UserEvent

### K.2.1 Event Filtering

One major deviation from the DVB-MHP specification is the addition of the Event Filter concept. In OCAP, the platform is required to allow the Monitor Application (if present) to intercept all qualified user input keycodes before they are received by an application. Qualified user input keycodes are those defined in Table 25–5 that are not identified as Mandatory Ordinary Keycodes.

**Note:** (Informative) Devices implementing OCAP may be multi-functional and may support other operating environments besides OCAP. The above requirements are true whenever the OCAP environment is 'selected' or any OCAP applications are presenting on a display or receiving input events.

To accomplish this, the Monitor Application installs a UserEventFilter. The UserEventFilter can:

- pass the event along unchanged
- modify the event keycode and then pass it along
- consume the event directly, without passing it along

Using this mechanism, the Monitor Application can perform actions such as changing the "focused" application before allowing the event to be processed. Alternatively, it could modify the keycode value before passing it on to the application.

The function setUserEventFilter() is restricted to use by applications which have the monitorapplication permission "filterUserEvents".

### K.2.2 Event Distribution

Events generated by the platform are distributed to applications based on the following distribution mechanism:

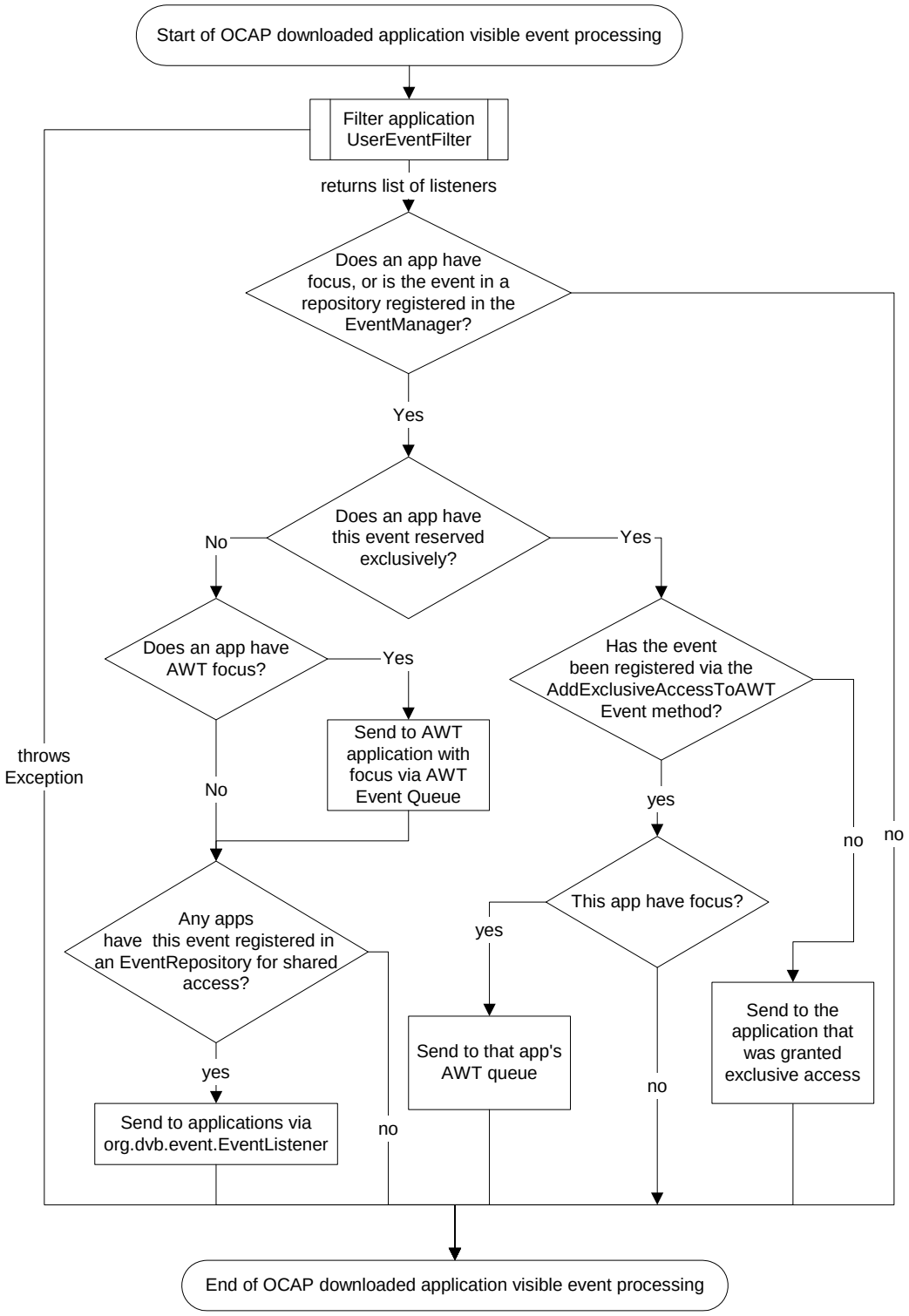



***Figure K–1 - The OCAP downloaded application visible event distribution mechanism***

### K.2.3 Event Resource Management

In OCAP, keys on a keyboard or remote control are considered "resources". In order to receive notification of an event occurrence, using the OCAP Event model, an application SHALL first "reserve" that "resource". By default, the application with focus has implicit reservation of all key events. Applications without focus may reserve key events using the OCAP EventManager. The OCAP EventManager derives from the DVB-MHP EventManager and implements the DAVIC resource model.

The EventManager is an object that is referenced using the `org.ocap.event.EventManager.getInstance()` method. This method SHALL return the OCAP EventManager instance.

### K.2.4 Event Listener

In order to receive notification of an event occurrence, an application registers an `org.dvb.event.UserEventListener` with the Event Manager using `org.ocap.event.EventManager.addUserEventListener`. `org.dvb.event.UserEventListener` is an interface which defines the method `userEventReceived()`. This method receives, as a parameter, the `org.ocap.event.UserEvent` which has occurred. An `org.dvb.event.UserEventListener` is removed with the method `org.dvb.event.EventManager.removeUserEventListener`.

Initially, user input events are limited to remote control and keyboard events. However, it is possible that new families of events (e.g., voice commands) may be defined in the future. User events that can be accessed by an application are defined in the `org.ocap.event.UserEvent` class.

#### *K.2.4.1 Event Repository*

To use this API, an application first creates an `org.dvb.event.UserEventRepository`, adds `UserEvents` to that repository, and then registers the repository with the Event Manager using `addUserEventListener`. At this time, the application specifies whether it wants exclusive access to these events (by specifying a `ResourceClient`), or whether it will share access to them. If the events are not already exclusively reserved by another application, the request is granted. After making this call, the repository may be deleted. If the application wishes to reserve additional events, it SHALL create a second repository to do so.

#### *K.2.4.2 Resource Client*

When an application wishes to gain exclusive access to a `UserEvent` it SHALL implement an `org.davic.resources.ResourceClient`. `ResourceClient` is an interface that defines methods for notification of an application when a second application attempts to gain exclusive access to that event. The three methods are:

| | |
|---|---|
| `releaseRequest()` | called to request that the client voluntarily give up the resource |
| `release()` | called to notify the client that the resource is being taken away |
| `releaseNotify()` | called to notify the client that the resource has been taken away |

#### *K.2.4.3 Resource Status Listener*

If an application can't get access to a resource, but wishes to be notified when the current owner gives up the resource, it SHALL implement an `org.davic.resource.ResourceStatusListener`. A `ResourceStatusListener` is registered with the Event Manager with the method

`EventManager.addResourceStatusEventListener`. This interface defines a method called by the Event Manager to notify listeners of a change in the status of a resource.

A `ResourceStatusListener` is removed with the method `EventManager.removeResourceStatusEventListener`.

#### *K.2.4.4 Example*

Example: request exclusive access to events for a non-focused application

```
import org.davic.resources.*  ;
import org.ocap.event.*;
import org.dvb.event.*;
import java.awt.event.*;

class ReserveMyKeyResources implements UserEventListener, ResourceStatusListener,
    ResourceClient {
    private int myStatus ;
    UserEventRepository repository ;

    public ReserveMyKeyResources() {
    org.ocap.event.EventManager em ;
    em = (org.ocap.event.EventManager)org.ocap.event.EventManager.getInstance () ;
    repository = new UserEventRepository ("R1") ;
    repository.addKey (KeyEvent.VK_NUMBER_SIGN) ;
    if (em.addUserEventListener ((UserEventListener)this, (ResourceClient)this,
            repository) == false) {
        em.addResourceStatusEventListener (this) ;
    }
    }

    /**
    * methods defined by the UserEventListener interface.
    */
    public void userEventReceived (org.dvb.event.UserEvent e) {
    ...
    }

    /**
    * Methods defined by the ResourceClient interface.
    */
    /**
    * In the case a cooperative application asks for an user event exclusively used by me.
    */
    public boolean requestRelease(ResourceProxy proxy, Object requestData) {
    String name ;
    // let's retrieve the name of the repository, that I have created, and
    // which contains the input event that the other application is asking for.
    name = ((org.dvb.event.RepositoryDescriptor)proxy).getName () ;
    if ((name.equals("R1")) && (myStatus == ...)) {
        // Ok, I release this event.
        return true ;
        } else {
        // No, I need this event, sorry !
        return false ;
    }
    }
    public void release (ResourceProxy proxy) {
    ...
    }
    public void notifyRelease (ResourceProxy proxy) {
    ...
    }
    public void statusChanged (ResourceStatusEvent event) {
    // find the source of the ResourceStatusEvent
    if(event instanceof UserEventAvailableEvent) {
        org.dvb.event.UserEventRepository availableRepository =
            (org.dvb.event.UserEventRepository)event.getSource();
        UserEvent availableEvent[] = availableRepository.getUserEvent();
        int i;
        for (i=0; i< availableEvent.length; i++) {
            if (availableEvent[i].getCode() == KeyEvent.VK_NUMBER_SIGN) {
```

```
                // try again to reserve the events in the repository
                em.addUserEventListener ((UserEventListener)this, (ResourceClient)this,
                    repository);
                break;
            }
        }
    }
    }
}
```

### K.2.5 Event Processing for multi-function devices

This section describes key event processing in multi-function implementations. For implementations that only contain an OCAP environment, no changes are required for key event processing.

#### *K.2.5.1 Description of Input Event Processing for Multi-Function Devices*

This section provides a description of the logical model for input event processing for Multi-Function Devices.

In OCAP, applications may receive input events either by reservation or by virtue of having input focus. Reservations may either be exclusive or shared; exclusive means that only one application at a time may reserve the event in question. See Section 3.1 for definition of Exclusively Reserved Keycode Event, Shared Reserved Keycode Event, and non-Reserved Keycode Event.

For the Multi-Function Device, a new class of event is introduced, the platform exclusive event – an event that is reserved permanently by a single application. OCAP has no means of assigning a permanent reservation, so this concept is only relevant to manufacturers' non-OCAP applications; for instance, a manufacturer may extend the required OCAP keycode set by adding a 'hot-button' that is always directed to a specific application. See Section 3.1 for definition of Platform Exclusive Keycode Event. Platform Exclusive Keycode Events SHALL always be delivered to the application that has reserved them, regardless of the state of the applications home environment or the mode of the applications.

Figure K–1 illustrates the conditions under which applications may receive these various events. App mode indicates the application mode, which is conditioned by the state of its home application environment.

***Table K–2 - Conditions under which applications receive events (Informative)***

| | App mode: Normal | App mode: background | App mode: cross-environment |
|---|---|---|---|
| Exclusive Reserved | Yes | Yes, if application retains reservation | Yes, if application retains reservation |
| Shared Reserved | Yes | Yes, if application retains reservation | Yes, if application retains reservation |
| Platform Reserved | Yes. Manufacturer apps only | Yes. Manufacturer apps only | Yes. Manufacturer apps only |
| non-Reserved | Yes, if application has input focus | No | Yes, if application has input focus |

Note that input events may be distributed to applications associated with different environments. Non-Reserved events SHALL be delivered to the application with input focus, whether its home environment is selected or it's presenting in cross-environment mode, and reserved events SHALL be delivered to all of the applications that currently have the event reserved, regardless of the state of its home environment.

### *K.2.5.2 Requirements for Event Processing for Multi-Function Devices*

Input event processing when cable is the selected environment is unchanged, that is, OCAP applications MAY receive input events via the AWT mechanism when they are the application in focus, and they MAY receive input events via the DVB event mechanism either as shared or exclusively reserved input events. Non-OCAP applications MAY receive input events when cable is the selected environment: either by virtue of having input focus while executing in cross-environment mode, or by virtue of having a reservation while executing in background mode.

When OCAP is not the currently selected environment, OCAP applications are either in background or cross environment mode.

When in background or cross environment mode, OCAP applications MAY receive Exclusively Reserved Keycode events reserved via the DVB event mechanism, however, there is no assurance that these key reservations will be respected by the rules and policies put into effect by the selected environment. If a reservation is lost, the implementation SHALL notify affected OCAP applications via the DAVIC Resource Client interface. In this case, OCAP applications are expected to re-establish their key event reservations when OCAP becomes the selected environment. The implementation SHALL grant key event reservation requests by OCAP applications, according to OCAP rules, when OCAP is the selected environment.

When in background or cross environment mode, OCAP applications MAY receive Shared Reserved Keycode events reserved via the DVB event mechanism, however, there is no assurance that these key events will be made available by the rules and policies put into effect by the selected environment. For those Shared Reserved Keycode events that will not be directed to OCAP applications, the implementation SHALL notify the applications via a UserEventUnavailableEvent. When OCAP becomes the selected environment, the implementation SHALL notify applications of the availability of these keys via the UserEventAvailableEvent.

When in cross-environment mode, OCAP applications MAY receive AWT events when they have input focus. The implementation SHALL distribute those input events to cross-environment mode applications in focus that the application would receive via AWT when OCAP is the selected environment and the application has focus.

#### *K.2.5.2.1 Input focus*

Input focus is an attribute of an application by which it receives non-Reserved Keycode Events, and may receive reserved events. For OCAP applications, events received while input focus is granted are via the AWT mechanism. For any combination of display and input devices, the implementation SHALL grant input focus to exactly one application at a time. On receipt of an input event, the implementation SHALL determine whether the event is exclusively reserved by any OCAP or non-OCAP application; if so, the event SHALL be directed to the application with the reservation, regardless of whether it has input focus or not.

#### *K.2.5.2.2 Multi-screen Scenarios*

This section describes requirements on those devices that have multi-screen capabilities, such as Picture in Picture (PiP) or Picture outside Picture (PoP). The following terms are used in this section:

PiP control keys - Manufacturer defined keys that are delivered exclusively to manufacturers' applications that control device specific capabilities. Keys may include PiP, PiP Move, PiP Swap, and PiP input select.

PiP application keys - The OCAP defined keys PiP Ch up and PiP Ch Down. These are delivered to applications whose environment is currently in the presenting state within a secondary event screen, defined below.

Primary event screen - In a multi-screen scenario, exactly one screen is the primary event screen at any one time. The application environment associated with this screen receives events in a normal manner, that is, as if no other screens were being displayed.

Secondary event screen - In a multi-screen scenario, exactly one screen is the secondary event screen at any one time. The application environment associated with this screen receives the PiP application keys.

The selected environment is associated with the primary event screen, such that the implementation SHALL distribute key events to applications associated with the primary event screen, in the fashion described in the preceding sections for the selected environment.

The application environment associated with the secondary event screen SHALL be in the presenting state. The PiP application key events SHALL be distributed to applications associated with the secondary event screen.

PiP control key events SHALL be distributed to manufacturer's applications. The manufacturer is responsible for allowing user's to configure primary, secondary, and other screens.

### Package org.ocap.event

| Interface Summary | |
|---|---|
| **UserEventFilter** | Only one instance of the class that implements this interface can be registered to EventManager via the EventManager.setUserEventFilter() method. |

| Class Summary | |
|---|---|
| **EventManager** | The event manager allows an application to receive events coming from the user. |
| **UserEvent** | Represents a user event. |
| **UserEventAction** | UserEventAction is returned by the UserEventFilter.filterUserEvent() method in order to inform the EventManager the value of the event and to which applications the event shall be forwarded. |

**org.ocap.event**

## Class EventManager

```
java.lang.Object
  └ org.dvb.event.EventManager
      └ org.ocap.event.EventManager
```

**All Implemented Interfaces:**

org.davic.resources.ResourceServer

```
public class EventManager
extends org.dvb.event.EventManager
```

The event manager allows an application to receive events coming from the user. These events can be sent exclusively to an application or can be shared between applications. The EventManager allows an application to ask for exclusive access to some events, these events being received either from the standard java.awt event mechanism or by the mechanism defined in this package. The EventManager is a singleton, and the instance is gotten from the getInstance() method. (Note that a type cast is necessary to gain reference to object of type org.ocap.event.EventManager.)

The right to receive events is considered as the same resource regardless of whether it is being handled exclusively or shared. An application successfully obtaining exclusive access to an event results in all other applications losing access to that event, whether the access of those applications was shared or exclusive.

If an UserEventFilter instance is set via EventManager.setUserEventFilter(), EventManager shall call the UserEventFilter.filterUserEvent() method before delivering events to the listening applications that are specified by the UserEventFilter. Note that EventManager shall call the filterUserEvent() method for only the events specified by an UserEventRepository instance which is set via EventManager.setFilteredRepository(). EventFilter may modify the key value of the userEvent and/or may direct the platform to forward the userEvent to a specific set of applications. Then EventManager gets the (possibly modified) event via UserEventAction.getEvent() and the list of AppIDs of the applications to receive the forwarded event via UserEventAction.getAppIDs(). EventManager shall call the UserEventListener.userEventReceived() of the applications which have the AppIDs specified by UserEventAction.getAppIDs(). For this purpose, EventManager shall track and keep the AppID of the applications which call the addExclusiveAccessToAWTEvent() and addUserEventListener() methods in a proprietary manner (manufacture dependent).

## Constructor Summary

| | |
|---|---|
| protected | **EventManager**()<br>Constructor for an instance of this class. |

## Method Summary

| | |
|---|---|
| org.dvb.event.UserEventRepository | **getFilteredRepository**()<br>Get the current UserEventRepository which specify the events to be filtered. |
| static org.dvb.event.EventManager | **getInstance**()<br>This method returns the sole instance of the org.ocap.event.EventManager class. |
| void | **setFilteredRepository**(org.dvb.event.UserEventRepository repository)<br>Sets the repository which specifies the events to be filtered. |
| void | **setUserEventFilter**(UserEventFilter filter)<br>Set the specified UserEventFilter to modify or consume the event and/or change the applications to deliver the event to. |

| Methods inherited from class org.dvb.event.EventManager |
|---|
| addExclusiveAccessToAWTEvent, addResourceStatusEventListener, addUserEventListener, addUserEventListener, removeExclusiveAccessToAWTEvent, removeResourceStatusEventListener, removeUserEventListener |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Constructor Detail

### EventManager

`protected` **`EventManager`**`()`

Constructor for an instance of this class. This constructor is provided for the use of implementations. 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.

# Method Detail

### getInstance

`public static org.dvb.event.EventManager` **`getInstance`**`()`

This method returns the sole instance of the org.ocap.event.EventManager class. The EventManager instance is either a singleton for each OCAP application or a singleton for a whole OCAP implementation. Note that a type cast is necessary for the return value.

**Returns:**

the instance of org.ocap.event.EventManager.

### getFilteredRepository

`public org.dvb.event.UserEventRepository` **`getFilteredRepository`**`()`

Get the current UserEventRepository which specify the events to be filtered. The monitorapplication permission is not necessary to call this method. This method is used to know which events are filtered at this moment. The UserEventRepository for event filtering is set via the setFilteredRepository() method.

**Returns:**

the current UserEventRepository which specifies the events to be filtered . EventManager maintains an empty UserEventRepository by default, and this is returned if setFilteredRepository() has not yet been called. If setFilteredRepository() has been called with a null UserEventRepository, then null is returned.

### setFilteredRepository

`public void`
**`setFilteredRepository`**`(org.dvb.event.UserEventRepository repository)`

Sets the repository which specifies the events to be filtered. Only one UserEventRepository instance can be set at a time. Multiple calls of this method will be result in an update of the UserEventReposiotry, i.e., the previous UserEventRepository is discarded and the new one is set. EventManager shall call the UserEventFilter.filterUserEvent() method only for the events specified by the UserEventRepository. By default, EventManager has an empty UserEventRepository, i.e., no UserEventFilter.filterUserEvent() method is called. The monitorapplication permission is necessary to call this method.

**Parameters:**

`repository` - a set of non-ordinary key events for calling the UserEventFilter.filterUserEvent() method. If null, the UserEventFilter.filterUserEvent() method is called for all events except the mandatory ordinary key events.

**Throws:**

`java.lang.SecurityException` - if the caller does not have monitorapplication permission("filterUserEvents") permission.

`java.lang.IllegalArgumentException` - if UserEventRepository contains Mandatory Ordinary keycodes.

### setUserEventFilter

`public void` **`setUserEventFilter`**`(UserEventFilter filter)`

Set the specified UserEventFilter to modify or consume the event and/or change the applications to deliver the event to. Only one UserEventFilter instance can be sent at a time. Multiple call of this method will result in update of the UserEventFilter, i.e., the previous UserEventFilter is discarded and the new one is set. By default, EventManager has no UserEventFilter (null). The monitorapplication permission is necessary to call this method.

**Parameters:**

`filter` - The filter to modify or consume the event and change the application to be delivered to.

**Throws:**

`java.lang.SecurityException` - if the caller does not have monitorapplication permission("filterUserEvents") permission.

**org.ocap.event**

## Class UserEvent

```
java.lang.Object
  └ java.util.EventObject
      └ org.dvb.event.UserEvent
          └ org.ocap.event.UserEvent
```

**All Implemented Interfaces:**

java.io.Serializable

```
public class UserEvent
extends org.dvb.event.UserEvent
```

Represents a user event. A user event is defined by a family, a type and either a code or a character. Unless stated otherwise, all constants used in this class are defined in `org.ocap.ui.event.OcRcEvent`, `java.awt.event.KeyEvent` and their parent classes.

# Field Summary

| Fields inherited from class org.dvb.event.UserEvent |
|---|
| UEF_KEY_EVENT |

| Fields inherited from class java.util.EventObject |
|---|
| source |

# Constructor Summary

| |
|---|
| **UserEvent**(java.lang.Object source, int family, char keyChar, long when)<br>Constructor for a new UserEvent object representing a key being typed. |
| **UserEvent**(java.lang.Object source, int family, int type, int code, int modifiers, long when)<br>Constructor for a new UserEvent object representing a key being pressed. |

# Method Summary

| | |
|---|---|
| void | **setCode**(int code)<br>Modifies the event code. |
| void | **setKeyChar**(char keychar)<br>Modifies the character associated with the key in this event. |

| Methods inherited from class org.dvb.event.UserEvent |
| --- |
| getCode, getFamily, getKeyChar, getModifiers, getType, getWhen, isAltDown, isControlDown, isMetaDown, isShiftDown |

| Methods inherited from class java.util.EventObject |
| --- |
| getSource, toString |

| Methods inherited from class java.lang.Object |
| --- |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

# Constructor Detail

### UserEvent

```
public UserEvent(java.lang.Object source,
                 int family,
                 int type,
                 int code,
                 int modifiers,
                 long when)
```

Constructor for a new UserEvent object representing a key being pressed.

**Parameters:**

`source` - the `EventManager` which is the source of the event.

`family` - the event family.

`type` - the event type. Either one of KEY_PRESSED or KEY_RELEASED.

`code` - the event code. One of the constants whose name begins in "VK_" defined in java.ui.event.KeyEvent, org.havi.ui.event or org.ocap.ui.event.OcRcEvent.

`modifiers` - the modifiers active when the key was pressed. These have the same semantics as modifiers in `java.awt.event.KeyEvent`.

`when` - a long integer that specifies the time the event occurred.

### UserEvent

```
public UserEvent(java.lang.Object source,
                 int family,
                 char keyChar,
                 long when)
```

Constructor for a new UserEvent object representing a key being typed. This is the combination of a key being pressed and then being released. The type of UserEvents created with this constructor shall be KEY_TYPED. Key combinations which do not result in characters, such as action keys like F1, shall not generate KEY_TYPED events.

**Parameters:**

`source` - the `EventManager` which is the source of the event

`family` - the event family.

`keyChar` - the character typed

`when` - a long integer that specifies the time the event occurred

**Since:**
MHP 1.0.1

# Method Detail

### setCode

`public void` **`setCode`**`(int code)`

Modifies the event code. For KEY_TYPED events, the code is VK_UNDEFINED.

**Throws:**
`java.lang.SecurityException` - if the caller does not have monitorapplication permission ("filterUserEvents").

**Since:**
OCAP 1.0

### setKeyChar

`public void` **`setKeyChar`**`(char keychar)`

Modifies the character associated with the key in this event. If no valid Unicode character exists for this key event, keyChar must be CHAR_UNDEFINED.

**Throws:**
`java.lang.SecurityException` - if the caller does not have monitorapplication permission ("filterUserEvents").

**Since:**
OCAP 1.0

**org.ocap.event**

## Class UserEventAction

```
java.lang.Object
  └ org.ocap.event.UserEventAction
```

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

UserEventAction is returned by the UserEventFilter.filterUserEvent() method in order to inform the EventManager the value of the event and to which applications the event shall be forwarded. See the org.ocap.event.UserEventFilter.filterUserEvent() method for further details. UserEventAction has separate methods to provide the list of AppIDs and the modified UserEvent instance. The modified UserEvent instance will be forwarded to the applications specified by AppIDs by EventManager. If the list of AppIDs is null, the EventManager shall forward the event to all registered UserEventListeners. If the list of AppIDs is not null, the EventManager shall forward the event to the registered UserEventListeners that match the AppIDs in the list. Note that if UserEventFilter.filterUserEvent() returns null, the event is not sent to any applications.

## Constructor Summary

| | |
|---|---|
| **UserEventAction**(UserEvent event, org.dvb.application.AppID[] appIDs)<br>Creates a UserEventAction instance. | |

## Method Summary

| | |
|---|---|
| org.dvb.application.AppID[] | **getAppIDs**()<br>Get the AppIDs to which the filtered event will be forwarded. |
| UserEvent | **getEvent**()<br>Get the event to be forwarded. |
| void | **setAppIDs**(org.dvb.application.AppID[] appIDs)<br>Sets the application IDs returned by this class. |
| void | **setEvent**(UserEvent event)<br>Sets the event returned by this class. |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

## Constructor Detail

### UserEventAction

```
public UserEventAction(UserEvent event,
```

```
org.dvb.application.AppID[] appIDs)
```

Creates a UserEventAction instance.

The event passed to this constructor SHOULD NOT be an application-defined subclass of UserEvent. If it is an application-defined subclass, then when the platform dispatches the event the platform MUST extract the parameters of the event (e.g., source, type, code etc.) and construct a new instance of the UserEvent class with those parameters. I.e., the EventManager MUST NOT deliver the application-defined subclass. (NOTE: This translation is done by the platform, NOT by this class).

**Parameters:**
`event` - The event to forward, or null for none.
`appIDs` - The AppIDs to which the filtered event will be forwarded, or null for default handling.

# Method Detail

### setEvent

```
public void setEvent(UserEvent event)
```

Sets the event returned by this class.

The event passed to this function SHOULD NOT be an application-defined subclass of UserEvent. If it is an application-defined subclass, then when the platform dispatches the event the platform MUST extract the data and construct a real UserEvent instance. (NOTE: This translation is done by the platform, NOT by this class.).

**Parameters:**
`event` - The event to forward, or null for none.

### setAppIDs

```
public void setAppIDs(org.dvb.application.AppID[] appIDs)
```

Sets the application IDs returned by this class.
**Parameters:**
`appIDs` - The AppIDs to which the filtered event will be forwarded, or null for default handling.

### getEvent

```
public UserEvent getEvent()
```

Get the event to be forwarded. The event may be modified while filtering. EventManager shall forward this modified event instead of the original user input event.
**Returns:**
The event to be forwarded. If null, no event is forwarded to any application.

### getAppIDs

```
public org.dvb.application.AppID[] getAppIDs()
```

Get the AppIDs to which the filtered event will be forwarded.
**Returns:**
The AppIDs to which the filtered event will be forwarded. If null, the EventManager shall forward the event to all registered UserEventListeners.

**org.ocap.event**

## Interface UserEventFilter

```
public interface UserEventFilter
```

Only one instance of the class that implements this interface can be registered to EventManager via the EventManager.setUserEventFilter() method. EventManager calls UserEventFilter.filterUserEvent() before a UserEvent is forwarded to any listening applications for events specified by the UserEventRepository set via the EventManager.setFilteredRepository() method.

# Method Summary

| | |
|---|---|
| UserEventAction | **filterUserEvent**(UserEvent e)<br>Called by the platform to filter user input events specified in the UserEventRepository when the event is received. |

# Method Detail

**filterUserEvent**

```
UserEventAction filterUserEvent(UserEvent e)
```

Called by the platform to filter user input events specified in the UserEventRepository when the event is received. The EventFilter may modify values in the UserEvent and/or change the applications to which the event is forwarded. The UserEventFilter.filterUserEvent() returns a UserEventAction instance which specifies the event to be forwarded and the list of AppIDs to which the event is forwarded. If it returns null, the event is consumed, i.e., the event is not forwarded to any applications.

**Parameters:**

e - the UserEvent which was received

**Returns:**

null if the event is to be terminated. Otherwise, a UserEventAction object that contains the event to distribute, and a list of AppIDs to which to distribute it.

# Annex L OCAP 1.1 Resource Management API

This section presents the `org.ocap.resource` APIs.

***Table L–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| Annex L OCAP 1.1 Resource Management API | No Corresponding Section | OCAP-Specific Extension |

## Package org.ocap.resource

org.ocap.resource The Resource Management API allows a monitor application to refuse a reservation of limited resources unconditionally and to resolve a resource reservation contention after negotiation.

| Interface Summary | |
|---|---|
| **ApplicationResourceUsage** | This interface represents a ResourceUsage corresponding to a resource explicitly reserved by an application my successfully calling one of the following OCAP calls: org.davic.mpeg.sections.SectionFilterGroup.attach(TransportStream, ResourceClient, Object) org.davic.net.tuning.NetworkInterfaceController.reserve(NetworkInterface, Object) org.davic.net.tuning.NetworkInterfaceController.reserveFor(Locator, Object) org.havi.ui.HBackgroundDevice.reserveDevice(ResourceClient) org.havi.ui.HGraphicsDevice.reserveDevice(ResourceClient) org.havi.ui.HVideoDevice.reserveDevice(ResourceClient) org.ocap.media.VBIFilterGroup.attach(ServiceContext serviceContext, ResourceClient client, Object requestData) An object implementing this interface should be used by the implementation to represent the ResourceUsage corresponding to a reserved resource when the ResourceContentionHandler.resolveResourceContention() method is invoked. |
| **ResourceContentionHandler** | A class implementing this interface decides which application shall be allowed to reserve a resource. |
| **ResourceUsage** | This interface represents a grouping of resources specific to an function performed by an application. |

| Class Summary | |
|---|---|
| **ResourceContentionManager** | This class manages a means of resolving a resource contention. |

## Package org.ocap.resource Description

## org.ocap.resource

The Resource Management API allows a monitor application to refuse a reservation of limited resources unconditionally and to resolve a resource reservation contention after negotiation. The monitor application can

implement a subclass of the `org.dvb.application.AppsDatabaseFilter` class to refuse a reservation, and a concrete class that implements the `org.ocap.resource.ResourceContentionHandler` interface to resolve a contention. See Section 19 Resource Management for more details.

### Example of Monitor Application

This sample code shows how the monitor application implements this package. The class ResourceHandler is one of the classes of the monitor application. It prevents an application that has an organization ID of REJECTED_ORGANIZATION from reserving a section filter resource, and gives a higher priority for resource reservation to an application that has an organization ID of PRIORITIZED_ORGANIZATION.

```
import org.ocap.resource.*;
import org.dvb.application.*;
import org.davic.resources.*;
import org.davic.mpeg.sections.*;

public class ResourceHandler extends AppsDatabaseFilter
        implements ResourceContentionHandler {

    private static final int REJECTED_ORGANIZATION = 0xABCD;
    private static final int PRIORITIZED_ORGANIZATION = 0x1234;

    /*
     * This is Constructor.
     * Set a ResourceFilter and a ResourceContentionManager for a resource
     * handling when constructing.
     */
    public ResourceHandler() {
        super();
        ResourceContentionManager rcManager =
ResourceContentionManager.getInstance();
        rcManager.setResourceFilter(this,
"org.davic.mpeg.sections.SectionFilterGroup");
        rcManager.setResourceContentionHandler(this);
    }

    /*
     * Check if the application is allowed to reserve a resource or not.
     */

    public boolean accept(AppID appid) {
        if(appid.getOID() == REJECTED_ORGANIZATION) {
            return(false);
        }
        return(true);
    }

    /*
     * Resolve a resource contention.
     */

     public ResourceUsage[] resolveResourceContention(
                        ResourceUsage newRequest,
                        ResourceUsage currentReservations[]) {
                ResourceUsage result[] = new
ResourceUsage[currentReservations.length + 1];
                if(newRequest.getAppID().getOID() == PRIORITIZED_ORGANIZATION)
{
                        result[0] = newRequest;
                        for(int i=0; i<currentReservations.length; i++) {
```

```
                                result[i+1] = currentReservations[i];
                        }
                } else {
                        for(int i=0; i<currentReservations.length; i++) {
                                result[i] = currentReservations[i];
                        }
                        result[currentReservations.length] = newRequest;
                }
                return(result);
        }
}

}
```

**org.ocap.resource**

## Interface ApplicationResourceUsage

**All Superinterfaces:**
ResourceUsage

```
public interface ApplicationResourceUsage
extends ResourceUsage
```

This interface represents a ResourceUsage corresponding to a resource explicitly reserved by an application my successfully calling one of the following OCAP calls:
org.davic.mpeg.sections.SectionFilterGroup.attach(TransportStream, ResourceClient, Object)
org.davic.net.tuning.NetworkInterfaceController.reserve(NetworkInterface, Object)
org.davic.net.tuning.NetworkInterfaceController.reserveFor(Locator, Object)
org.havi.ui.HBackgroundDevice.reserveDevice(ResourceClient)
org.havi.ui.HGraphicsDevice.reserveDevice(ResourceClient)
org.havi.ui.HVideoDevice.reserveDevice(ResourceClient) org.ocap.media.VBIFilterGroup.attach(ServiceContext serviceContext, ResourceClient client, Object requestData) An object implementing this interface should be used by the implementation to represent the ResourceUsage corresponding to a reserved resource when the ResourceContentionHandler.resolveResourceContention() method is invoked.

# Method Summary

| Methods inherited from interface org.ocap.resource.ResourceUsage |
| --- |
| getAppID, getResource, getResourceNames |

**org.ocap.resource**

## Interface ResourceContentionHandler

```
public interface ResourceContentionHandler
```

A class implementing this interface decides which application shall be allowed to reserve a resource.

An application which has a MonitorAppPermission("handler.resource") may have a class implementing this interface, and may set an instance of it in the ResourceContentionManager. The `resolveResourceContention(org.ocap.resource.ResourceUsage, org.ocap.resource.ResourceUsage[])` method decides the how to resolve resource conflicts between the new request and existing resource allocations. See the ResourceContentionManager for the details.

# Method Summary

| | |
|---|---|
| `ResourceUsage[]` | **`resolveResourceContention`**`(ResourceUsage newRequest, ResourceUsage[] currentReservations)`<br>This method notifies the ResourceContentionHandler that one to many resource contentions have occurred between one or more applications and system modules, except the Emergency Alert System (EAS) module. |
| `void` | **`resourceContentionWarning`**`(ResourceUsage newRequest, ResourceUsage[] currentReservations)`<br>Warns the resource contention handler of an impending contention with a presenting ServiceContext (e.g., scheduled recording as defined by the OCAP DVR specification). |

# Method Detail

### resolveResourceContention

```
ResourceUsage[] resolveResourceContention(ResourceUsage newRequest,
                                          ResourceUsage[] currentReservations)
```

This method notifies the ResourceContentionHandler that one to many resource contentions have occurred between one or more applications and system modules, except the Emergency Alert System (EAS) module. EAS system module resource requests SHALL be given the highest priority by the implementation and resource requests by this module SHALL not be reported to the ResourceContentionHandler. In the case of one application, the same application is conflicting with itself and a registered ResourceContentionHandler SHALL be notified in this case.

This method notifies the ResourceContentionHandler that one to many resource contentions have occurred between two or more applications. Each entry in the currentReservations indicates a set of resources reserved by an application for a single activity such as a resource usage by a single service context. There may be multiple entries in this list from a single application. An entry may correspond to a current resource usage or resource reservations for a future activity. A prioritized array of `ResourceUsage` instances is returned. The array is in priority order from highest to lowest indicating the priority order to be followed by the implementation while resolving the conflicts. When this method returns the implementation will iterate through each entry in the array in the order of priority, awarding resources as required by the activity represented by the resourceUsage. The ResourceContentionHandler may use information such as Application priority to prioritize the array of ResourceUsages returned. When the value returned is not null the ResourceContentionHandler MAY return an array containing all of the `ResourceUsage` objects passed to it, or it MAY return a subset of those objects.

**Parameters:**
`newRequest` - The resource usage object containing the attributes of the resource[s] request.
`currentReservations` - The resource usage objects currently owned by applications which are in conflict with the newRequest. A ResourceUsage associated with a current reservation may belong to an application that has been destroyed. Use of the AppID contained within such a ResourceUsage with any of the methods in `org.dvb.application.AppsDatabase` MAY cause a failure status to be returned.
**Returns:**
A prioritized array of resource usage objects. The first entry has the highest priority. This function returns null if the contention handler wants the implementation to resolve the conflict.

### resourceContentionWarning

```
void resourceContentionWarning(ResourceUsage newRequest,
                               ResourceUsage[] currentReservations)
```

Warns the resource contention handler of an impending contention with a presenting ServiceContext (e.g., scheduled recording as defined by the OCAP DVR specification). If a ResourceContentionHandler is registered the implementation SHALL call this method as defined by the `ResourceContentionManager.setWarningPeriod(int)` method.
**Parameters:**
`newRequest` - The resource usage object containing the attributes of the resource[s] request.
`currentReservations` - The resource usage objects currently owned by applications which are in conflict with the newRequest. A ResourceUsage associated with a current reservation may belong to an application that has been destroyed. Use of the AppID contained within such a ResourceUsage with any of the methods in `org.dvb.application.AppsDatabase` MAY cause a failure status to be returned.

**org.ocap.resource**

## Class ResourceContentionManager

```
java.lang.Object
  └ org.ocap.resource.ResourceContentionManager
```

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

This class manages a means of resolving a resource contention.

An application which has a MonitorAppPermission ("handler.resource") may have a subclass of the AppsDatabaseFilter class or a class implementing the ResourceContentionHandler interface, and may set an instance of them in the ResourceContentionManager. The concrete class of the AppsDatabaseFilter class identifies an application that is not allowed absolutely to reserve the resource. The class implementing the ResourceContentionHandler interface resolves a resource contention after a resource negotiation.

See the section 19 Resource Management in this specification for details.

## Constructor Summary

| | |
|---|---|
| protected | **ResourceContentionManager**()<br>A constructor of this class. |

## Method Summary

| | |
|---|---|
| static ResourceContentionManager | **getInstance**()<br>This method returns an instance of the ResourceContentionManager class. |
| int | **getWarningPeriod**()<br>Gets the warning period set by the setWarningPeriod method. |
| void | **setResourceContentionHandler**(ResourceContentionHandler handler)<br>This method sets the specified ResourceContentionHandler that decides which application shall be denied reserving a scarce resource. |
| void | **setResourceFilter**(org.dvb.application.AppsDatabaseFilter filter, java.lang.String resourceProxy)<br>This method sets an instance of a concrete class that extends AppsDatabaseFilter. |
| void | **setWarningPeriod**(int warningPeriod)<br>Sets the warning period used by the implementation to determine when to call the resourceContentionWarning method in a registered ResourceContentionHandler. |

| Methods inherited from class java.lang.Object |
| --- |
| `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait` |

# Constructor Detail

### ResourceContentionManager

`protected` **`ResourceContentionManager`**`()`

A constructor of this class. An application must use the `getInstance()` method to create an instance.

# Method Detail

### getInstance

`public static ResourceContentionManager` **`getInstance`**`()`

This method returns an instance of the ResourceContentionManager class. It is not required to be a singleton manner.

**Returns:**

The ResourceContentionManager instance.

### setResourceFilter

`public void` **`setResourceFilter`**`(org.dvb.application.AppsDatabaseFilter filter, java.lang.String resourceProxy)`

This method sets an instance of a concrete class that extends AppsDatabaseFilter. The AppsDatabaseFilter.accept(AppID) method returns true if an application specified by the AppID is allowed to reserve the resource, and returns false if the application is not allowed to reserve it. At most, only one AppsDatabaseFilter is set for each type of resource. Multiple calls of this method replace the previous instance by a new one. If an AppsDatabaseFilter has not been associated with the resource, then any application is allowed to reserve the resource. By default, no AppsDatabaseFilter is set, i.e., all applications are allowed to reserve the resource.

**Parameters:**

`filter` - the AppsDatabaseFilter to deny the application reserving the specified resource. If null is set, the AppsDatabaseFilter for the specified resource will be removed.

`resourceProxy` - A full path class name of a concrete class of the org.davic.resources.ResourceProxy interface. It specifies a resource type that the specified AppsDatabaseFilter filters. For example, "org.davic.net.tuning.NetworkInterfaceController".

**Throws:**

`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("handler.resource").

### setResourceContentionHandler

`public void` **`setResourceContentionHandler`**`(ResourceContentionHandler handler)`

This method sets the specified ResourceContentionHandler that decides which application shall be denied reserving a scarce resource. At most only one instance of ResourceContentionHandler can be set. Multiple calls of this method replace the previous instance by a new one. By default, no

ResourceContentionHandler is set, i.e. the `ResourceContentionHandler.resolveResourceContention(org.ocap.resource.ResourceUsage, org.ocap.resource.ResourceUsage[])` method is not called.

**Parameters:**

`handler` - the ResourceContentionHandler to be set. If null is set, the ResourceContentionHandler instance will be removed and the `ResourceContentionHandler.resolveResourceContention(org.ocap.resource.ResourceUsage, org.ocap.resource.ResourceUsage[])` method will not be called.

**Throws:**

`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("handler.resource").

### getWarningPeriod

```
public int getWarningPeriod()
```

Gets the warning period set by the setWarningPeriod method.

**Returns:**

The warning period in milli-seconds.

### setWarningPeriod

```
public void setWarningPeriod(int warningPeriod)
                     throws java.lang.IllegalArgumentException
```

Sets the warning period used by the implementation to determine when to call the resourceContentionWarning method in a registered `ResourceContentionHandler`. If the parameter is zero the implementation SHALL NOT call the resourceContentionWarning method. If the parameter is non-zero the implementation SHALL call the resourceContentionWarning method if it has enough information to do so. Setting the warningPeriod to non-zero MAY NOT cause the resourceContentionWarning method to be called for two reasons, 1) the implementation cannot determine when contention is going to happen, and 2) the warning period is longer than the duration to the contention.

**Parameters:**

`warningPeriod` - New warning period in milli-seconds. If the value is smaller than the minimum clock resolution supported by the implementation, the implementation MAY round it up to the minimum.

**Throws:**

`java.lang.IllegalArgumentException` - if the parameter is negative.

`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("handler.resource").

**org.ocap.resource**

## Interface ResourceUsage

**All Known Subinterfaces:**

ApplicationResourceUsage, ServiceContextResourceUsage

```
public interface ResourceUsage
```

This interface represents a grouping of resources specific to an function performed by an application.

# Method Summary

| | |
|---|---|
| `org.dvb.application.AppID` | **`getAppID`**`()`<br>Gets the `AppID` of the application associated with the set of resources contained in the usage. |
| `org.davic.resources.ResourceProxy` | **`getResource`**`(java.lang.String resourceName)`<br>Gets the instance of `ResourceProxy` corresponding to a resource name returned by the getResourceNames method of this ResourceUsage. |
| `java.lang.String[]` | **`getResourceNames`**`()`<br>Gets the array of resource names associated with the resource reservation. |

# Method Detail

### getAppID

```
org.dvb.application.AppID getAppID()
```

Gets the `AppID` of the application associated with the set of resources contained in the usage.

**Returns:**

AppID of the application.

### getResourceNames

```
java.lang.String[] getResourceNames()
```

Gets the array of resource names associated with the resource reservation.

**Returns:**

The array of qualified java class names for the resources used (or required) for this resource usage.

### getResource

```
org.davic.resources.ResourceProxy getResource(java.lang.String resourceName)
```

Gets the instance of `ResourceProxy` corresponding to a resource name returned by the getResourceNames method of this ResourceUsage. This method will return null if the resource is not yet reserved. This method provide information to distinguish which resource in the ResourceUsage has already

been reserved. Since DAVIC resource API reserves resource in one by one manner, the ResourceUsage may include both reserved and unreserved resources.

**Parameters:**

`resourceName` - The fully qualified java name for a resource included in this ResourceUsage.

**Returns:**

The instance of `ResourceProxy` corresponding to the resourceName.

**Throws:**

`java.lang.IllegalArgumentException` - if the resourceName is not in the list of fully qualified java class names returned by the method getResourceNames()

# Annex M Streamed Media API Extensions

The streamed media APIs provide a consistent interface for streaming digital content such as audio, video, and ancillary data.

These extensions allow applications to get information associated with the format and aspect ratio of the video being presented to the user. This API also provides controls for setting and querying the video presentation.

## M.1 DVB-GEM and DVB-MHP Specification Correspondence

This section extends Annex N of the [DVB-GEM 1.1] and [DVB-MHP1.1.2]. The OpenCable Application Platform is in complete compliance with the following classes and interfaces from the `org.dvb.media` API:

***Table M–1 - DVB-GEM Specification Compliance***

| org.dvb.media Class |
|---|
| ActiveFormatDescriptionChangedEvent |
| AspectRatioChangedEvent |
| BackgroundVideoPresentationControl |
| CAException |
| CAStopEvent |
| DFCChangedEvent |
| DripFeedDataSource |
| DripFeedPermission |
| NoComponentSelectedEvent |
| PresentationChangedEvent |
| ServiceRemovedEvent |
| StopByResourceLossEvent |
| VideoFormatEvent |
| VideoFormatListener |

## M.2 Deviations from the DVB-MHP Specification

The information in this section deviates from the corresponding section in the [DVB-MHP1.1.2]. The OpenCable Application Platform deviates from the following classes and interfaces in the `org.dvb.media` API:

***Table M–2 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and DVB-MHP 1.0***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| Annex M Streamed Media API Extensions | Annex N (normative): Streamed media API extensions | Extension | Annex N (normative): Streamed Media API Extensions | Extension |

### M.2.1 org.dvb.media.VideoFormatControl

#### *M.2.1.1 Fields*

##### *M.2.1.1.1 DFC_PROCESSING_CCO Field*

public static final int DFC_PROCESSING_CCO

A 4:3 central part out of the 640x480 input 16:9 frame is transferred into a 640x480 4:3 output frame.

##### *M.2.1.1.2 DFC_PROCESSING_FULL Field*

public static final int DFC_PROCESSING_FULL

The full 640x480 frame is transferred. (This may be either 4:3 or 16:9;part of this may be black (e.g., in the "pillar box" cases.)

##### *M.2.1.1.3 DFC_PROCESSING_LB_14_9 Field*

public static final int DFC_PROCESSING_LB_14_9

The 640x480 input grid is transferred into a 14:9 LB in a 4:3 frame.

##### *M.2.1.1.4 DFC_PROCESSING_LB_16_9 Field*

public static final int DFC_PROCESSING_LB_16_9

The 640x480 input grid is transferred into a 16:9 letterbox in a 4:3 frame.

##### *M.2.1.1.5 DFC_PROCESSING_LB_2_21_1_ON_16_9 Field*

public static final int DFC_PROCESSING_LB_2_21_1_ON_16_9

The 640x480 input grid is transferred into a 2.21:1 letterbox in a 16:9 frame.

##### *M.2.1.1.6 DFC_PROCESSING_LB_2_21_1_ON_4_3 Field*

public static final int DFC_PROCESSING_LB_2_21_1_ON_4_3

The 640x480 input grid is transferred into a 2.21:1 letterbox in a 4:3 frame.

##### *M.2.1.1.7 DFC_PROCESSING_PAN_SCAN Field*

public static final int DFC_PROCESSING_PAN_SCAN

A 4:3 part out of the 640x480 input 16:9 or 2.21:1 frame is transferred into a 640x480 4:3 output.

frame. The horizontal position of this part is determined by pan and scan vectors from the MPEG video stream.

#### *M.2.1.2 Methods*

##### *M.2.1.2.1 getActiveFormatDefinition()*

This definition is compliant with the definition of getActiveFormatDefinition() in section N.1 of [DVB-GEM 1.1], with the following extension:

References to ETR 154 should be replaced with references to ANSI/SCTE 43 [SCTE 43].

### M.2.2 org.dvb.media.VideoPresentationControl

#### *M.2.2.1 Methods*

##### *M.2.2.1.1 getClipRegion()*

This definition is compliant with the definition of getClipRegion() in the [DVB-MHP1.1.2] except for the following extension:

References to ETR 154 should be replaced with references to [SCTE 54].

##### *M.2.2.1.2 getInputVideoSize()*

public java.awt.Dimension getInputVideoSize()

This method returns the dimensions of the video before any scaling has taken place (but after [SCTE 54] up-sampling). On 50Hz standard definition systems, this method always returns 640x480.

Returns:

the size of the decoded video before any scaling has taken place (but after [SCTE 54] up-sampling)

##### *M.2.2.1.3 setClipRegion(Rectangle)*

This definition is compliant with the definition of getClipRegion() in the [DVB-MHP1.1.2] except for the following extension:

References to ETR 154 should be replaced with references to [SCTE 54].

### M.2.3 org.dvb.media.VideoTransformation

#### *M.2.3.1 Constructors*

##### *M.2.3.1.1 VideoTransformation(Rectangle,float,float,HScreenPoint)*

This definition is compliant with the definition of getClipRegion() in the [DVB-MHP1.1.2] except for the following extension:

References to ETR 154 should be replaced with references to [SCTE 54].

#### *M.2.3.2 Methods*

##### *M.2.3.2.1 getClipRegion()*

This definition is compliant with the definition of getClipRegion() in the [DVB-MHP1.1.2] except for the following extension:

References to ETR 154 should be replaced with references to [SCTE 54].

#### *M.2.3.2.2 setClipRegion(Rectangle)*

This definition is compliant with the definition of getClipRegion() in the [DVB-MHP1.1.2] except for the following extension:

References to ETR 154 should be replaced with references to [SCTE 54].

## M.3 OCAP 1.1 Specific Requirements

This information extends the specification requirements made to the [DVB-MHP1.1.2].

An OCAP 1.1 implementation SHALL be capable of presenting DTVCC contents for MPEG programs.

# Annex N Application Listing and Launching

This section of the OCAP 1.1 Profile provides an API for managing the life cycle of a service bound application. This API also provides access to various information about registered applications.

## N.1 DVB-GEM and DVB-MHP Specification Compliance

This section corresponds to Annex S of the [DVB-MHP1.1.2] and [DVB-GEM 1.1]. The OpenCable Application Platform is in complete compliance with the API, org.dvb.application, specified in this section.

***Table N–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| Annex N Application Listing and Launching | Annex S | Extension | Annex S | Extension |

## N.2 Deviations from the DVB-MHP Specification

There are no deviations from the corresponding section in the [DVB-GEM 1.1].

## N.3 OCAP 1.1 Specific Requirements

There are no additional OCAP 1.1 Profile requirements to [DVB-MHP1.1.2].

# Annex O OCAP 1.1 Fundamental Classes API

This section provides OCAP fundamental functions.

## O.1 DVB-GEM and DVB-MHP Specification Correspondence

This section of the OCAP 1.1 Profile corresponds to Annex I of [DVB-MHP1.1.2] and [DVB-GEM 1.1] as follows:

***Table O–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| Annex O OCAP 1.1 Fundamental Classes API | Annex I (normative): DVB-J fundamental classes | Extension | Annex I (normative): DVB-J fundamental classes | Extension |

## O.2 OCAP 1.1 Specific Requirements

There are no deviations from Annex I of [DVB-MHP1.1.2] and [DVB-GEM 1.1].

### O.2.1 Extensions to DVB-MHP (Normative)

This information extends the specification requirements made to the [DVB-MHP1.1.2] and [DVB-GEM 1.1].

The following information is normative for the OCAP 1.1 Profile. It is specific to OCAP 1.1 and is not contained within the [DVB-MHP1.1.2] or [DVB-GEM 1.1].

#### *O.2.1.1 org.ocap package*

The `org.ocap` package provides the following fundamental OCAP classes:

### Package org.ocap

| Class Summary | |
|---|---|
| **OcapSystem** | This class provides system utility functions. |

**org.ocap**

### Class OcapSystem

```
java.lang.Object
  └ org.ocap.OcapSystem

public final class OcapSystem
extends java.lang.Object
```

This class provides system utility functions.

# Method Summary

| | |
|---|---|
| `static void` | **`monitorConfiguredSignal`**`()`<br>Called by the Initial Monitor Application to inform the OCAP implementation it has completed its configuration process and that the boot processing may resume. |
| `static void` | **`monitorConfiguringSignal`**`(int port, int timeout)`<br>Called by the monitor application to inform the OCAP implementation that it is configuring and the boot process may resume after it calls the `monitorConfiguredSignal` method, see Section 20.2.2.2 Boot Process while connected to the cable network – CableCARD device present. |

| **Methods inherited from class java.lang.Object** |
|---|
| `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait` |

# Method Detail

### monitorConfiguringSignal

```
public static void monitorConfiguringSignal(int port,
                                            int timeout)
                                     throws java.io.IOException
```

Called by the monitor application to inform the OCAP implementation that it is configuring and the boot process may resume after it calls the `monitorConfiguredSignal` method, Section 20.2.2.2 Boot Process while connected to the cable network – CableCARD device present.
On invocation of this method, the APIs used for conformance testing, specifically, `org.ocap.test.OCAPTest` SHALL be initialized for use. This means that the implementation SHALL perform the following actions:

- a. Open a socket for receiving UDP datagrams on a port, the value of which is specified in the `port` parameter passed to this method.
- b. Wait to receive a datagram that contains a string formatted thus: `ate:a.b.c.d:xxxx:ppp` (string may be null-terminated), where 'a.b.c.d' represents an IPv4 address, and 'xxxx' represents an IP port number, and 'ppp' represents protocol type ('TCP' for TCP/IP and 'UDP' for UDP/IP). Any received datagrams which do not contain a properly formatted payload string SHALL be ignored. Once a datagram with a properly formatted string has been received, the datagram socket SHALL be closed.
- c. Attempt to establish a TCP or UDP socket connection to the test system using the IPv4 address and port number obtained in b. The protocol type for the socket connection is specified by 'ppp' string in b. This connected socket SHALL be used solely to transmit and receive data originating from the `org.ocap.test.OCAPTest` APIs and SHALL NOT be accessible to applications through other APIs. The TCP or UDP socket connection shall have a timeout of 'infinite'. If this method does not complete within the specified timeout period, an `IOException` SHALL be thrown.
- d. Return control to the caller.

If this method is called with both the `port` and `timeout` parameters set to 0, then the OCAP implementation SHALL not enable the conformance testing APIs, which SHALL just return silently, without performing any action.
If the monitor application does not call this method in the time specified in section 20.2.2.3 Boot Process while connected to the cable network - CableCARD device present, then the OCAP implementation SHALL behave the same as if this method had been called with 0 specified for both the `port` and `timeout` parameters. If the monitor application does not call this method in the time specified in section 20.2.2.3 Boot Process while connected to the cable network - CableCARD device present, then the implementation SHALL behave the same as if this method had been called with 0 specified for both the `port` and `timeout` parameters.

**Parameters:**

`port` - the IP port number to listen for datagrams from the test system on.

`timeout` - the time, in seconds to allow for a communications channel to be established with the test system.

**Throws:**

`java.lang.SecurityException` - if application does not have MonitorAppPermission("signal configured").

`java.io.IOException` - if a communications channel cannot be established with the test system within the amount of time specified by the `timeout` parameter.

### monitorConfiguredSignal

`public static void` **`monitorConfiguredSignal`**`()`

Called by the Initial Monitor Application to inform the OCAP implementation it has completed its configuration process and that the boot processing may resume. It is recommended that the monitor call this method as soon as possible after the `monitorConfiguringSignal` method has been called.

# Annex P OCAP 1.1 Service API

***Table P–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| Annex P OCAP 1.1 Service API | No Corresponding Section | OCAP-Specific Extension |

## Package org.ocap.service

| Interface Summary | |
|---|---|
| **AbstractService** | The AbstractService represents a non-broadcast Service which applications can be added to and be selected as a group within a given javax.tv.service.selection.ServiceContext. |
| **ServiceContextResourceUsage** | This interface represents a ResourceUsage corresponding to a group of resources implicitly reserved by the implementation for the successful completion of the ServiceContext.select() method. |

| Class Summary | |
|---|---|
| **AbstractServiceType** | This class represents the abstract service type value. |
| **ServiceTypePermission** | `ServiceTypePermission` represents application permission to select a specific service type using a `ServiceContext` accessible by the application. |

### org.ocap.service
### Interface AbstractService

**All Superinterfaces:**
javax.tv.service.Service

```
public interface AbstractService
extends javax.tv.service.Service
```

The AbstractService represents a non-broadcast Service which applications can be added to and be selected as a group within a given javax.tv.service.selection.ServiceContext.

Note the following subinterface-specific behavior for methods defined by the javax.tv.service.Service superinterface:

- The hasMultipleInstances() method shall always return false.
- The getServiceType() method shall always return AbstractServiceType.OCAP_ABSTRACT_SERVICE.

# Method Summary

| | |
|---|---|
| java.util.Enumeration | **getAppAttributes**()<br>Returns a java.util.Enumeration of references to org.ocap.application.OcapAppAttributes instances. |
| java.util.Enumeration | **getAppIDs**()<br>Returns a java.util.Enumeration of references to org.dvb.application.AppID instances. |

| **Methods inherited from interface javax.tv.service.Service** |
|---|
| equals, getLocator, getName, getServiceType, hashCode, hasMultipleInstances, retrieveDetails |

# Method Detail

### getAppIDs

java.util.Enumeration **getAppIDs**()

Returns a java.util.Enumeration of references to org.dvb.application.AppID instances. These application IDs correspond to applications that are associated with this abstract service.

**Returns:**

the java.util.Enumeration of application IDs included in this AbstractService

### getAppAttributes

java.util.Enumeration **getAppAttributes**()

Returns a java.util.Enumeration of references to org.ocap.application.OcapAppAttributes instances. These application attributes correspond to applications that are associated with this abstract service.

**Returns:**

the java.util.Enumeration of application attributes for applications included in this AbstractService.

**org.ocap.service**

## Class AbstractServiceType

```
java.lang.Object
  └ javax.tv.service.ServiceType
      └ org.ocap.service.AbstractServiceType
```

```
public class AbstractServiceType
extends javax.tv.service.ServiceType
```

This class represents the abstract service type value.

# Field Summary

| | |
|---|---|
| `static javax.tv.service.ServiceType` | **`OCAP_ABSTRACT_SERVICE`**<br>ServiceType for an abstract service. |

| **Fields inherited from class javax.tv.service.ServiceType** |
|---|
| `ANALOG_RADIO, ANALOG_TV, DATA_APPLICATION, DATA_BROADCAST, DIGITAL_RADIO, DIGITAL_TV, NVOD_REFERENCE, NVOD_TIME_SHIFTED, UNKNOWN` |

# Constructor Summary

| | |
|---|---|
| `protected` | **`AbstractServiceType`**`(java.lang.String name)`<br>Provides an instance of AbstractServiceType. |

# Method Summary

| **Methods inherited from class javax.tv.service.ServiceType** |
|---|
| `toString` |

| **Methods inherited from class java.lang.Object** |
|---|
| `clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait` |

# Field Detail

**OCAP_ABSTRACT_SERVICE**

`public static final javax.tv.service.ServiceType` **`OCAP_ABSTRACT_SERVICE`**
ServiceType for an abstract service.

# Constructor Detail

### AbstractServiceType

`protected` **`AbstractServiceType`**`(java.lang.String name)`
Provides an instance of AbstractServiceType.
**Parameters:**
`name` - The string name of this type (i.e. "OCAP_ABSTRACT_SERVICE").

**org.ocap.service**

## Interface ServiceContextResourceUsage

**All Superinterfaces:**
ResourceUsage

```
public interface ServiceContextResourceUsage
extends ResourceUsage
```

This interface represents a ResourceUsage corresponding to a group of resources implicitly reserved by the implementation for the successful completion of the ServiceContext.select() method. An object implementing this interface should be used by the implementation to represent ResourceUsages corresponding to ServiceContext when the ResourceContentionHandler.resolveResourceContention() method is invoked.

# Method Summary

| | |
|---|---|
| javax.tv.service.Service | **getRequestedService**()<br>Gets the Service that was requested when the resource contention was incurred. |
| javax.tv.service.selection.ServiceContext | **getServiceContext**()<br>Gets the ServiceContext for which the resources have been reserved. |

| Methods inherited from interface org.ocap.resource.ResourceUsage |
|---|
| getAppID, getResource, getResourceNames |

# Method Detail

### getServiceContext

javax.tv.service.selection.ServiceContext **getServiceContext**()
Gets the ServiceContext for which the resources have been reserved.
**Returns:**
the ServiceContext for which the resources have been reserved.

### getRequestedService

javax.tv.service.Service **getRequestedService**()
Gets the Service that was requested when the resource contention was incurred.
**Returns:**
Service requested.

**org.ocap.service**
## Class ServiceTypePermission

```
java.lang.Object
  └ java.security.Permission
      └ java.security.BasicPermission
          └ org.ocap.service.ServiceTypePermission
```

**All Implemented Interfaces:**
java.io.Serializable, java.security.Guard

```
public final class ServiceTypePermission
extends java.security.BasicPermission
```

`ServiceTypePermission` represents application permission to select a specific service type using a `ServiceContext` accessible by the application.

When this permission is evaluated, the `SecurityManager.checkPermission` method must not fail when checking for `SelectPermission` on the accessed `ServiceContext`. Otherwise, the security manager check for this permission will also fail.

Note that undefined service type strings may be provided to the constructor of this class, but subsequent calls to `SecurityManager.checkPermission()` with the resulting `ServiceTypePermission` object will fail.

## Field Summary

| | |
|---|---|
| `static java.lang.String` | **`BROADCAST`**<br>Indicates an inband broadcast service provided by a content provider. |
| `static java.lang.String` | **`MFR`**<br>Indicates an abstract service provided by the Host device manufacturer. |
| `static java.lang.String` | **`MSO`**<br>Indicates an abstract service provided by the HFC network provider (i.e., MSO). |

## Constructor Summary

| |
|---|
| **`ServiceTypePermission`**`(java.lang.String type, java.lang.String actions)`<br>Creates a new ServiceTypePermission object with the specified service type name. |

## Method Summary

| | |
|---|---|
| `boolean` | **`equals`**`(java.lang.Object obj)`<br>Tests two `ServiceTypePermission` objects for equality. |
| `java.lang.String` | **`getActions`**`()`<br>Returns the canonical representation of the actions string. |

# Method Summary

| | |
|---|---|
| int | **hashCode**()<br>Provides the hash code value of this object. |
| boolean | **implies**(java.security.Permission p)<br>Checks if the specified permission is "implied" by this object. |

| Methods inherited from class java.security.BasicPermission |
|---|
| newPermissionCollection |

| Methods inherited from class java.security.Permission |
|---|
| checkGuard, getName, toString |

| Methods inherited from class java.lang.Object |
|---|
| clone, finalize, getClass, notify, notifyAll, wait, wait, wait |

# Field Detail

### MFR

```
public static final java.lang.String MFR
```
Indicates an abstract service provided by the Host device manufacturer.

### MSO

```
public static final java.lang.String MSO
```
Indicates an abstract service provided by the HFC network provider (i.e., MSO).

### BROADCAST

```
public static final java.lang.String BROADCAST
```
Indicates an inband broadcast service provided by a content provider.

# Constructor Detail

### ServiceTypePermission

```
public ServiceTypePermission(java.lang.String type,
                             java.lang.String actions)
```
Creates a new ServiceTypePermission object with the specified service type name.

**Parameters:**

`type` - The name of the service type that can be selected. Supported service types include "abstract.manufacturer", "abstract.mso", and "broadcast". An asterisk may be used to signify a wildcard match.
`actions` - The actions String is either "own" or "*". The string "own" means the permission applies to your own service context, acquired via the `ServiceContextFactory.createServiceContext` or `ServiceContextFactory.getServiceContext` methods. The string "*" implies permission to these, plus permission for service contexts obtained from all other sources.

# Method Detail

### implies

`public boolean` **`implies`**`(java.security.Permission p)`

Checks if the specified permission is "implied" by this object.

Specifically, implies(Permission p) returns true if:

- *p* is an instance of ServiceTypePermission and
- *p*'s action string matches this object's, or this object or *p* has "*" as an action string, and
- *p*'s type string matches this object's, or this object has "*" as a type string.

In addition, implies(Permission p) returns true if:

- *p* is an instance of SelectPermission and,
- *p's* locator contains an actual or implied source_id value which corresponds to the type string in this object where ISO/IEC 13818-1 defines broadcast source_id values that correspond to a broadcast type string and table 11-7 defines abstract service values that correspond to abstract MSO and abstract manufacturer type strings.
- *p's* action string matches this object's, or this object has "*" as an action string.

**Overrides:**
`implies` in class `java.security.BasicPermission`
**Parameters:**
`p` - The permission against which to test.
**Returns:**
`true` if the specified permission is equal to or implied by this permission; `false` otherwise.

### equals

`public boolean` **`equals`**`(java.lang.Object obj)`

Tests two `ServiceTypePermission` objects for equality. Returns `true` if and only if `obj`'s class is the same as the class of this object, and `obj` has the same name and actions string as this object.
**Overrides:**
`equals` in class `java.security.BasicPermission`
**Parameters:**
`obj` - The object to test for equality.
**Returns:**
`true` if the two permissions are equal; `false` otherwise.

### hashCode

`public int` **`hashCode`**`()`

Provides the hash code value of this object. Two `ServiceTypePermission` objects that are equal will return the same hash code.

**Overrides:**
`hashCode` in class `java.security.BasicPermission`
**Returns:**
The hash code value of this object.

### getActions

`public java.lang.String` **`getActions`**`()`

Returns the canonical representation of the actions string.
**Overrides:**
`getActions` in class `java.security.BasicPermission`
**Returns:**
The actions string of this permission.

# Annex Q OCAP 1.1 System API

This section presents the `org.ocap.system` APIs. This API is used to implement assumable system modules described in Section 20.2.4.11.

***Table Q–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| Annex Q OCAP 1.1 System API | No Corresponding Section | OCAP-Specific Extension |

# Example for Privileged Application assumption of EAS Audio Presentation and Setting of EAS Display Attributes

This is a sample code for an assuming OCAP-J Private Host Application. See also Section 20.2.4.11.1.

```
import org.ocap.system.*;
public class PrivateHostApp implements SystemModuleHandler{
SystemModule            sm; // A SystemModule returns by ready() method.
    SystemModuleRegistrar   registrar;
// Note that this is not an actual byte data.
byte[] privateHostAppID = {0x01, 0x23, 0x45, 0x67};
/**
* Constructor of this class.
* Register this class as an assuming Private Host Application.
     */
    public PrivateHostApp() {
registrar = SystemModuleRegistrar.getInstance();
// Note that a current resident Private Host Application
// (ID=0x01234567) will terminate by this method call.
registrar.registerSASHandler(this, privateHostAppID);
}

/**
* Unregister this class, i.e., terminate assuming.
*/
public void unregister() {
//Unregister itself.
registrar.unregisterSASHandler(privateHostAppID);
}

/**
* Define the receiveAPDU() method in the SystemModuleHandler.
*/
public void receiveAPDU(int apduTag, int lengthField, byte[] dataByte) {
      // Define a process for each receiving APDU here.
}/**
* Define the sendAPDUFailed() method in the SystemModuleHandler.
*/
public void sendAPDUFailed(int apduTag, byte[] dataByte) {
       // Define an error recovery for each receiving APDU here.
       // For example, re-send it.
       sm.sendAPDU(apduTag, dataByte);
    }

    /**
* Define the notifyUnregister() method in the SystemModuleHandler.
*/
public void notifyUnregister() {
       // Terminate activity and communication with the POD.
       // Return immediately!
       // Write a termination procedure here.
}

    /**
* Define the ready() method in the SystemModuleHandler.
*/
public void ready(SystemModule systemModule) {
       // Note that this is not an actual byte data.
       int    sasDataRqstApduTag = 0x9F9A02;
       byte[] sas_data_rqst_byte = {0x01, 0x23, 0x45, 0x67};
```

```
            //Start communication with the POD.
sm = systemModule;
sm.sendAPDU(sasDataRqstApduTag, sas_data_rqst_byte);
}
}
```

## Example of an assuming MMI module (Monitor Application)

This is sample code for an assuming OCAP-J MMI module. See also Section 20.2.4.11.2.

```
import org.ocap.system.*;
import org.ocap.hardware.pod.*;
public class MMI implements SystemModuleHandler{
SystemModule            sm; // A SystemModule returns by ready() method.
SystemModuleRegistrar   registrar;

    /**
* Constructor of this class.
* Register this class as an assuming MMI.
*/
public MMI() {
       registrar = SystemModuleRegistrar.getInstance();
            // Note that a resident MMI doesn't terminate by this method call.
            registrar.registerMMIHandler(this);
POD pod = POD.getInstance();
PODApplication[] apps = pod.getApplications();
String url = apps[0].getURL();
}

    /**
* Unregister this class, i.e., terminate assuming.
*/
public void unregister() {
      //Unregister itself.
      registrar.unregisterMMIHandler();
    }

    /**
     * Define the receiveAPDU() method in the SystemModuleHandler.
*/
public void receiveAPDU(int apduTag, int lengthField, byte[] dataByte) {
       // Define a process for each receiving APDU here.
       // For example, draw a new MMI dialogue.
    }

    /**
* Define the sendAPDUFailed() method in the SystemModuleHandler.
*/
public void sendAPDUFailed(int apduTag, byte[] byteData) {
      // Define a error recovery for each receiving APDU here.
      // For example, re-send it.
      sm.sendAPDU(apduTag, byteData);
    }

    /**
* Define the notifyUnregister() method in the SystemModuleHandler.
*/
public void notifyUnregister() {
      // Terminate activity and communication with the POD.
      // Return immediately!
      // Write a termination procedure here.
    }

    /**
* Define the ready() method in the SystemModuleHandler.
*/
public void ready(SystemModule systemModule) {
// Note that this is not an actual byte data.
int    serverQueryApduTag = 0x9F8022;
byte[] server_query_byte = {0x12, 0x34, 0x56};
//Start communication with the POD.
sm = systemModule;
sm.sendAPDU(serverQueryApduTag, server_query_byte);
    }
}
```

## Example for Privileged Application assumption of EAS Audio Presentation and Setting of EAS Display Attributes

This is a sample code for OCAP-J application assumption of EAS module audio presentation and setting of EAS display attributes. See also Section 20.2.4.11.4.

```
import org.ocap.system.*;

/**
 * This is a sample implementation of EAS assuming module.
 */
public class EASModule implements EASHandler {
    /**
     * Constructor of this class.
     * The EASmodule class is a part of the Monitor Application. It SHALL
     * have a MonitorAppPermission("handler.eas") permission.
     */
    public EASModule() {
        try {
            EASModuleRegistrar easmr = EASModuleRegistrar.getInstance();
            easmr.registerEASHandler(this);

            //Investigate possible font colors.
            java.awt.Color fontColor[] =
                    (java.awt.Color[])easmr.getEASCapability(
                                EASModuleRegistrar.EAS_ATTRIBUTE_FONT_COLOR);

            //Investigate possible font styles.
            String fontStyle[] =
                    (String[])easmr.getEASCapability(
                                EASModuleRegistrar.EAS_ATTRIBUTE_FONT_STYLE);

            //Investigate possible font face.
            String fontFace[] =
                    (String[])easmr.getEASCapability(
                                EASModuleRegistrar.EAS_ATTRIBUTE_FONT_FACE);

            //Set a preferred font color/style/face.
            //(Set the first item for every attribute as a sample.
            //An actual Monitor Application may provide a GUI to select
            //user preferences.)
            int attributes[] = {EASModuleRegistrar.EAS_ATTRIBUTE_FONT_COLOR,
                                EASModuleRegistrar.EAS_ATTRIBUTE_FONT_STYLE,
                                EASModuleRegistrar.EAS_ATTRIBUTE_FONT_FACE};
            Object values[] = {fontColor[0], fontStyle[0], fontFace[0]};
            easmr.setEASAttribute(attributes, values);

        }
        catch (Exception e) {
        }
    }

/**
     * Defined in EASHandler.
     */
    public boolean notifyPrivateDescriptor(byte[] descriptor) {
        // Play audio according to the descriptor.
        return(true);
    }

/**
     * Defined in EASHandler.
     */
    public void stopAudio() {
       //Stop audio that was informed by notifyPrivateDescriptor() method.
    }
}
```

### Package org.ocap.system

| Interface Summary | |
|---|---|
| **EASHandler** | An OCAP-J application can register an EASHandler to the EASModuleRegistrar via the `EASModuleRegistrar.registerEASHandler` |

| | |
|---|---|
| | `(org.ocap.system.EASHandler)` method. |
| **EASListener** | This interface represents a listener that will receive EAS events. |
| **SystemModule** | The SystemModule is used by an OCAP-J application that has a role of an assuming module to send an APDU to the CableCARD device. |
| **SystemModuleHandler** | The SystemModuleHandler is used by an OCAP-J application that has a role of an assuming module for the following purposes: 1) receive an APDU from the CableCARD device, 2) detect an unsent APDU to the POD in case of an error, and 3) notification of becoming registered and unregistered. |

| Class Summary | |
|---|---|
| **EASEvent** | This class represents an EAS event. |
| **EASManager** | This class represents a manager that allows applications to register listeners for EAS events. |
| **EASModuleRegistrar** | An OCAP-J application can set and get a preferred attribute value of an EAS alert text. |
| **MonitorAppPermission** | The MonitorAppPermission class represents permission to execute privileged operations only Monitor Application should be granted. |
| **RegisteredApiManager** | This class represents a manager for registered APIs that can be registered with an implementation by a privileged application. |
| **RegisteredApiUserPermission** | The RegisteredApiUserPermission class represents permission for an application to use a specific registered API. |
| **SystemModuleRegistrar** | This class is used by an OCAP-J application to assume the system modules. |

**org.ocap.system**

## Class EASEvent

```
java.lang.Object
  └ java.util.EventObject
      └ org.ocap.system.EASEvent
```

**All Implemented Interfaces:**

java.io.Serializable

```
public class EASEvent
extends java.util.EventObject
```

This class represents an EAS event.

| Field Summary | |
|---|---|
| `static int` | **`EAS_COMPLETE`**<br>The event is generated when an EAS message completes. |

# Field Summary

| | |
|---|---|
| static int | **EAS_DETAILS_CHANNEL**<br>This event is generated when an EAS table is detected by the implementation and the table requires a forced tune. |
| static int | **EAS_TEXT_DISPLAY**<br>This event is generated when an EAS table is detected by the implementation and the table requires textual display. |

| Fields inherited from class java.util.EventObject |
|---|
| source |

# Constructor Summary

| |
|---|
| **EASEvent**(java.lang.Object source, int reason)<br>Constructs a EASEvent with the specified source and reason. |

# Method Summary

| | |
|---|---|
| int | **getReason**()<br>Gets the reason passed to the constructor. |

| Methods inherited from class java.util.EventObject |
|---|
| getSource, toString |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

# Field Detail

### EAS_DETAILS_CHANNEL

public static final int **EAS_DETAILS_CHANNEL**

This event is generated when an EAS table is detected by the implementation and the table requires a forced tune.

### EAS_TEXT_DISPLAY

public static final int **EAS_TEXT_DISPLAY**

This event is generated when an EAS table is detected by the implementation and the table requires textual display.

#### EAS_COMPLETE

```
public static final int EAS_COMPLETE
```

The event is generated when an EAS message completes.

# Constructor Detail

#### EASEvent

```
public EASEvent(java.lang.Object source,
                int reason)
```

Constructs a EASEvent with the specified source and reason.

**Parameters:**

`source` - Implementation specific source object

`reason` - The reason that caused this event. See constants in this class for possible reasons.

# Method Detail

#### getReason

```
public int getReason()
```

Gets the reason passed to the constructor.

**Returns:**

Reason for this event.

**org.ocap.system**

## Interface EASHandler

```
public interface EASHandler
```

An OCAP-J application can register an EASHandler to the EASModuleRegistrar via the `EASModuleRegistrar.registerEASHandler(org.ocap.system.EASHandler)` method. The `notifyPrivateDescriptor(byte[])` of this class is called to notify a location of an alternative audio for EAS representation. The OCAP-J application can play an audio specified by a private descriptor.

**See Also:**
`EASModuleRegistrar`

# Method Summary

| | |
|---|---|
| `boolean` | **`notifyPrivateDescriptor`**`(byte[] descriptor)`<br>This is a call back method to notify a private descriptor in the cable_emergency_alert message defined in SCTE 18 2001. |
| `void` | **`stopAudio`**`()`<br>This is a call back method to notify that the alert duration has finished. |

# Method Detail

### notifyPrivateDescriptor

```
boolean notifyPrivateDescriptor(byte[] descriptor)
```

This is a call back method to notify a private descriptor in the cable_emergency_alert message defined in SCTE 18 2001. If the alert_priority=15 but no audio specified by SCTE 18 2001 is available, the OCAP implementation shall call this method. The OCAP-J application can get a location of an alternative audio specified in the private descriptor and play it according to SCTE 18 2001. If the OCAP-J application doesn't support the private descriptor, the EAShandler.notifyPrivateDescriptor() method shall return false and the OCAP implementation can play detailed channel or proprietary audio. This method shall return immediately. The audio shall be played in a unique thread not to prevent an alert text representation.

**Parameters:**
`descriptor` - an array of bytes of a private descriptor in the cable_emergency_alert message defined in SCTE 18 2001.
**Returns:**
true if the OCAP-J application can sound an audio of the location of the specified descriptor.

### stopAudio

```
void stopAudio()
```

This is a call back method to notify that the alert duration has finished. The OCAP-J application stops an audio specified by a private descriptor. The OCAP-J application shall not unregister the EASHandler until terminating an audio by this method.

**org.ocap.system**

## Interface EASListener

**All Superinterfaces:**
java.util.EventListener

```
public interface EASListener
extends java.util.EventListener
```

This interface represents a listener that will receive EAS events.

# Method Summary

| | |
|---|---|
| void | **notify**(EASEvent event)<br>Notifies an application that an EAS message has started, is in-progress, or has completed. |
| void | **warn**(EASEvent event)<br>Warns an application that an EAS message has been received and parsed and resources will soon be allocated to it. |

# Method Detail

### warn

```
void warn(EASEvent event)
```

Warns an application that an EAS message has been received and parsed and resources will soon be allocated to it. The application can get the reason for the event from the event parameter.

**Parameters:**
`event` - EAS event received.

### notify

```
void notify(EASEvent event)
```

Notifies an application that an EAS message has started, is in-progress, or has completed. The application can get the reason for the event from the event parameter.

**Parameters:**
`event` - EAS event received.

**org.ocap.system**

## Class EASManager

```
java.lang.Object
  └ org.ocap.system.EASManager

public abstract class EASManager
extends java.lang.Object
```

This class represents a manager that allows applications to register listeners for EAS events.

## Field Summary

| | |
|---|---|
| static int | **EAS_MESSAGE_IN_PROGRESS_STATE**<br>Indicates the implementation is processing the EAS message and EAS information is being presented. |
| static int | **EAS_MESSAGE_RECEIVED_STATE**<br>Indicates an EAS message has been received and is about to be processed. |
| static int | **EAS_NOT_IN_PROGRESS_STATE**<br>Indicates an EAS message is not being processed. |

## Constructor Summary

| |
|---|
| **EASManager**() |

## Method Summary

| | |
|---|---|
| abstract void | **addListener**(EASListener listener)<br>Adds a listener for EAS events. |
| static EASManager | **getInstance**()<br>Gets the instance of the EAS Manager class that may be used by the application to register an EASListener. |
| int | **getState**()<br>Gets the EAS state. |
| abstract void | **removeListener**(EASListener listener)<br>Removes a listener from receiving EAS events. |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Field Detail

### EAS_MESSAGE_RECEIVED_STATE

`public static final int` **`EAS_MESSAGE_RECEIVED_STATE`**

Indicates an EAS message has been received and is about to be processed. This state MAY be entered before an EASListener receives an EAS event.

### EAS_MESSAGE_IN_PROGRESS_STATE

`public static final int` **`EAS_MESSAGE_IN_PROGRESS_STATE`**

Indicates the implementation is processing the EAS message and EAS information is being presented. This state MAY coincide with resources being taken away from applications.

### EAS_NOT_IN_PROGRESS_STATE

`public static final int` **`EAS_NOT_IN_PROGRESS_STATE`**

Indicates an EAS message is not being processed. This state MAY be entered before an EASListener receives an EAS_COMPLETE_EVENT.

# Constructor Detail

### EASManager

`public` **`EASManager`**`()`

# Method Detail

### getInstance

`public static EASManager` **`getInstance`**`()`

Gets the instance of the EAS Manager class that may be used by the application to register an EASListener.

**Returns:**

The EAS manager.

### addListener

`public abstract void` **`addListener`**`(EASListener listener)`

Adds a listener for EAS events.

**Parameters:**

`listener` - The new EAS listener.

### removeListener

`public abstract void` **`removeListener`**`(EASListener listener)`

Removes a listener from receiving EAS events. If the parameter listener wasn't previously added with the `addListener` method, this method does nothing.

**Parameters:**
`listener` - The EAS listener to be removed.

### getState

`public int` **`getState`**`()`

Gets the EAS state. Possible return values are defined by state constants in this class.
**Returns:**
EAS state.

**org.ocap.system**

## Class EASModuleRegistrar

```
java.lang.Object
  └ org.ocap.system.EASModuleRegistrar
```

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

An OCAP-J application can set and get a preferred attribute value of an EAS alert text. The capability method provides possible attribute values. And also, an OCAP-J application can set an EASHandler to be notified that a private descriptor indicating an alternative audio, if the alert_priority=15 but no audio specified by SCTE 18 2001 is available. The OCAP implementation may throw an exception if a specified preference doesn't conform to FCC rules or the SCTE 18 specification. For example, some color combinations could make text unreadable. See http://www.fcc.gov/eb/eas/ for FCC rules.

See also Section 20 *Baseline Functionality* for detailed assuming mechanism.

**Since:**

1.0

## Field Summary

| | |
|---|---|
| `static int` | **`EAS_ATTRIBUTE_BACK_COLOR`**<br>Indicates a background color attribute of an EAS alert text. |
| `static int` | **`EAS_ATTRIBUTE_BACK_OPACITY`**<br>Indicates a background opacity attribute of an EAS alert text. |
| `static int` | **`EAS_ATTRIBUTE_FONT_COLOR`**<br>Indicates a font color attribute of an EAS alert text. |
| `static int` | **`EAS_ATTRIBUTE_FONT_FACE`**<br>Indicates a font face attribute of an EAS alert text. |
| `static int` | **`EAS_ATTRIBUTE_FONT_OPACITY`**<br>Indicates a font opacity attribute of an EAS alert text. |
| `static int` | **`EAS_ATTRIBUTE_FONT_SIZE`**<br>Indicates a font size attribute of an EAS alert text. |
| `static int` | **`EAS_ATTRIBUTE_FONT_STYLE`**<br>Indicates a font style attribute of an EAS alert text. |

## Constructor Summary

| | |
|---|---|
| `protected` | **`EASModuleRegistrar()`**<br>A constructor of this class. |

## Method Summary

| | |
|---|---|
| java.lang.Object | **getEASAttribute**(int attribute)<br>This method returns a current attribute values applied to an EAS alert text on a screen. |
| java.lang.Object[] | **getEASCapability**(int attribute)<br>This method returns a possible attribute values applied to an EAS alert text on a screen. |
| static EASModuleRegistrar | **getInstance**()<br>This method returns a sole instance of the EASModuleRegistrar class. |
| void | **registerEASHandler**(EASHandler handler)<br>This method registers an EASHandler instance. |
| void | **setEASAttribute**(int[] attribute, java.lang.Object[] value)<br>This method sets a preferred attribute values applied to an EAS alert text on a screen. |
| void | **unregisterEASHandler**()<br>This method unregisters the current registered EASHandler instance. |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

## Field Detail

### EAS_ATTRIBUTE_FONT_COLOR

public static final int **EAS_ATTRIBUTE_FONT_COLOR**

Indicates a font color attribute of an EAS alert text.

### EAS_ATTRIBUTE_FONT_STYLE

public static final int **EAS_ATTRIBUTE_FONT_STYLE**

Indicates a font style attribute of an EAS alert text.

### EAS_ATTRIBUTE_FONT_FACE

public static final int **EAS_ATTRIBUTE_FONT_FACE**

Indicates a font face attribute of an EAS alert text.

### EAS_ATTRIBUTE_FONT_SIZE

public static final int **EAS_ATTRIBUTE_FONT_SIZE**

Indicates a font size attribute of an EAS alert text.

#### EAS_ATTRIBUTE_BACK_COLOR

`public static final int` **`EAS_ATTRIBUTE_BACK_COLOR`**

Indicates a background color attribute of an EAS alert text.

#### EAS_ATTRIBUTE_FONT_OPACITY

`public static final int` **`EAS_ATTRIBUTE_FONT_OPACITY`**

Indicates a font opacity attribute of an EAS alert text.

#### EAS_ATTRIBUTE_BACK_OPACITY

`public static final int` **`EAS_ATTRIBUTE_BACK_OPACITY`**

Indicates a background opacity attribute of an EAS alert text.

# Constructor Detail

#### EASModuleRegistrar

`protected` **`EASModuleRegistrar`**`()`

A constructor of this class. An application must use the `getInstance()` method to create an instance.

# Method Detail

#### getInstance

`public static EASModuleRegistrar` **`getInstance`**`()`

This method returns a sole instance of the EASModuleRegistrar class. The EASModuleRegistrar instance is either a singleton for each OCAP application or a singleton for an entire OCAP implementation.

**Returns:**

a singleton EASModuleRegistrar instance.

**Throws:**

`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("handler.eas").

#### registerEASHandler

`public void` **`registerEASHandler`**`(EASHandler handler)`

This method registers an EASHandler instance. At most only one EASHandler instance can be registered. Multiple calls of this method replace the previous instance by a new one. By default, no instance is registered. If null is specified, this method do nothing and raise an exception.

**Throws:**

`java.lang.IllegalArgumentException` - if null is specified.

#### unregisterEASHandler

`public void` **`unregisterEASHandler`**`()`

This method unregisters the current registered EASHandler instance. If no EASHandler instance has registered, do nothing.

### getEASCapability

```
public java.lang.Object[] getEASCapability(int attribute)
```

This method returns a possible attribute values applied to an EAS alert text on a screen. Note that the possible font attribute may be different from the possible font for Java application since the EAS may be implemented by native language.

**Parameters:**
`attribute` - one of constants that has a prefix of EAS_ATTRIBUTE_ to specify an attribute to get possible values.
**Returns:**
an array of possible attribute values of an alert text corresponding to the specified attribute parameter.

- If the attribute parameter is EAS_ATTRIBUTE_FONT_COLOR, an array of java.awt.Color that represents possible font color returns. The Color.getString() shall return a text expression of its color to show a user.
- If the attribute parameter is EAS_ATTRIBUTE_FONT_STYLE, an array of String that represents possible font style returns.
- If the attribute parameter is EAS_ATTRIBUTE_FONT_FACE, an array of String that represents possible font face name returns.
- If the attribute parameter is EAS_ATTRIBUTE_FONT_SIZE, an array of Integer of possible font size returns.
- If the attribute parameter is EAS_ATTRIBUTE_BACK_COLOR, an array of java.awt.Color that represents possible background color returns. The Color.getString() shall return a text expression of its color to show a user.

**Throws:**
`java.lang.IllegalArgumentException` - if the attribute is out of range.

### setEASAttribute

```
public void setEASAttribute(int[] attribute,
                            java.lang.Object[] value)
```

This method sets a preferred attribute values applied to an EAS alert text on a screen. If the specified attribute value is invalid, i.e., the value is not included in the return value of the `getEASCapability(int)` method, this method doesn't change current attribute value and throw an exception. If the specified attribute is EAS_FONT_OPACITY or EAS_BACK_OPACITY, this method accepts any Float value and the OCAP implementation tries to apply it. The OCAP implementation may not able to apply the specified opacity exactly.

Note that even if the application could set a preference successfully, the OCAP implementation may not apply it to an EAS message text on a screen if a conflict with FCC rule or SCTE 18 specification is found while displaying process.

**Parameters:**
`attribute` - an array of constants that has a prefix of EAS_ATTRIBUTE_ to specify an attribute.
`value` - an array of preferred attribute values to be set to an alert text. The i-th item of the value array corresponds to the i-th item of the attribute array.

- If the attribute parameter is EAS_ATTRIBUTE_FONT_COLOR, an instance of java.awt.Color that represents preferred font color.
- If the attribute parameter is EAS_ATTRIBUTE_FONT_STYLE, an String that represents preferred font style.
- If the attribute parameter is EAS_ATTRIBUTE_FONT_FACE, an String that represents preferred font face name.
- If the attribute parameter is EAS_ATTRIBUTE_FONT_SIZE, an Integer of preferred font size.
- If the attribute parameter is EAS_ATTRIBUTE_BACK_COLOR, an instance of java.awt.Color that represents preferred background color.
- If the attribute parameter is EAS_ATTRIBUTE_FONT_OPACITY, an Float of preferred font opacity.
- If the attribute parameter is EAS_ATTRIBUTE_BACK_OPACITY, an Float of preferred background opacity.

**Throws:**
`java.lang.IllegalArgumentException` - if the attribute is out of range or the value is not a possible value or if the specified preference conflicts with FCC rules or SCTE 18. For example, an EAS message is invisible since same color is specified to a font and background. Criteria of visibility depends on a manufacturer or a display device etc.

### getEASAttribute

```
public java.lang.Object getEASAttribute(int attribute)
```

This method returns a current attribute values applied to an EAS alert text on a screen.

**Parameters:**
`attribute` - one of constants that has a prefix of EAS_ATTRIBUTE_ to specify an attribute to get current values.
**Returns:**
a current attribute values of an alert text corresponding to the specified attribute parameter.

- If the attribute parameter is EAS_ATTRIBUTE_FONT_COLOR, an instance of java.awt.Color that represents current font color returns.
- If the attribute parameter is EAS_ATTRIBUTE_FONT_STYLE, an String that represents current font style returns.
- If the attribute parameter is EAS_ATTRIBUTE_FONT_FACE, an String that represents current font face name returns.
- If the attribute parameter is EAS_ATTRIBUTE_FONT_SIZE, an Integer of current font size returns.
- If the attribute parameter is EAS_ATTRIBUTE_BACK_COLOR, an instance of java.awt.Color that represents current background color returns.
- If the attribute parameter is EAS_ATTRIBUTE_FONT_OPACITY, an Float of current font opacity returns.

- If the attribute parameter is EAS_ATTRIBUTE_BACK_OPACITY, an Float of current background opacity returns.

**Throws:**
`java.lang.IllegalArgumentException` - if the attribute is out of range.

**org.ocap.system**

## Class MonitorAppPermission

```
java.lang.Object
  └ java.security.Permission
      └ java.security.BasicPermission
          └ org.ocap.system.MonitorAppPermission
```

**All Implemented Interfaces:**

java.io.Serializable, java.security.Guard

```
public final class MonitorAppPermission
extends java.security.BasicPermission
```

The MonitorAppPermission class represents permission to execute privileged operations only Monitor Application should be granted.

A MonitorAppPermission consists of a permission name, representing a single privileged operation. The name given in the constructor may end in ".*" to represent all permissions beginning with the given string, such as `"*"` to allow all MonitorAppPermission operations, or `"handler.*"` to only allow setting any handler.

The following table lists all MonitorAppPermission permission names.

| Permission Name | What the Permission Allows | Description |
|---|---|---|
| registrar | Provides access to the Application Database by way of the AppRegistrar | This permission allows the caller to add or remove applications from the Application Database. |
| handler.registrar | Set an application signal handler via the AppManagerProxy.setAppSignalHandler() method. | This facility allows Monitor Applications to reject an updated XAIT, in which case the applications database and stored applications are not changed. |
| service | Allows creation of an AbstractService | Applications with this permission can create and manage their own service contexts and the services running in those service contexts. |
| servicemanager | Allows management of all services | Applications with this permission can create their own service contexts and manage both their own and other service contexts and services. |
| security | Allows setting the `SecurityPolicyHandler` used by the AppManager | This permission allows the application to register a SecurityPolicyHandler with the AppManager to determine the PermissionCollection granted to applications before they are run. |
| reboot | Initiates a system to reboot itself | This permission allows the caller to request for a system reboot. |
| systemevent | Allows setting the error, resource depletion, or reboot handlers | This permission allows the Monitor Application to be alerted upon system reboot, resource depletion, and error events. |
| handler.appFilter | Set a new black and white list to the system | This permission allows the application to set a new black and white list, which the system uses to determine whether to accept or reject broadcasted applications on the receiver. Such control should only be granted to a monitor application. |

| Permission Name | What the Permission Allows | Description |
|---|---|---|
| handler.resource | Set a Resource Contention Handler | Set a handler to resolve resource contention between two or more apps see `ResourceContentionManager.setResourceContentionHandler(org.ocap.resource.ResourceContentionHandler)`. |
| handler.closedCaptioning | Set closed-captioning preferences and control captioning. | Allows monitor application to get a `ClosedCaptioningAttribute` and call methods in a `ClosedCaptioningControl`. |
| handler.analogAudioControl | Set Primary or Secondary audio. | Allows monitor application to call methods in a `AnalogAudioControl`. |
| filterUserEvents | Filter user events | This permission allows the user to filter user events. |
| handler.eas | Set preferences of Emergency Alert System (EAS) message representation. | Allows monitor application to set preferences of EAS message representation and add a new EAShandler by calling `EASModuleRegistrar.registerEASHandler(org.ocap.system.EASHandler)`. |
| setVideoPort | Allows enabling and disabling video port | Allows monitor to call org.ocap.hardware.VideoOutputPort.enable() and org.ocap.hardware.VideoOutputPort.disable(). |
| podApplication | Allows assuming a Private Host Application Specific Application Support Resource | Allows Monitor Application to call org.ocap.system.SystemModuleRegistrar. |
| signal.configured | Allows monitor to signal implementation to resume boot processing after handlers have been set | Allows monitor to call org.ocap.OcapSystem.monitorConfiguredSignal(). |
| storage | Provides control of persistent storage devices and content stored therein | Allows monitor to delete volumes it does not own, initialize `StorageProxy` associated devices, make detachable devices ready for detaching or ready for use, and set file access permissions for any application accessible file or directory. |
| properties | Allows monitor to access ocap system properties | Allows monitor to call read ocap properties that require monitor application permission. |
| registeredapi | Provides access to network specific APIs | Gives monitor ability to register an API, remove a registered API, or access a registered API |
| vbifiltering | Allows monitor application to filter VBI data. | Allows monitor application to call a VBIFilterGroup constructor. |
| codeDownload | Allows monitor application to initiate a download, generally following a CVT signaling a deferred download | Allow monitor application to call Host.codeDownload method |
| mediaAccess | Allows monitor application to register MediaAccessHandler. | Allows monitor application to call a MediaAccessHandlerRegistrar.setExternalTriggers(). Allows monitor application to call a MediaAccessConditionControl.conditionHasChanged(). |
| quarantine | Allows monitor application to remove applications from quarantine. | Allows monitor application to call removeQuarantinedApplication(). |

| Permission Name | What the Permission Allows | Description |
|---|---|---|
| testApplications | Allows monitor application to control how applications with "test_application_flag" set are handled | Allows monitor application to control whether applications with "test_application_flag" are ignored (as normal) or whether this flag is ignored. This is controlled by the method AppManagerProxy.enableTestApplications(). |
| environment.selection | Allows monitor application to request the cable environment become .* selected or deselected. | Allows monitor application to call request the cable environment .* become selected or deselected by calling Environment.select or .* Environment.deselect |
| diagnostics | Allows monitor application the get the MIB manager. | |

Other permissions may be added as necessary.

# Constructor Summary

**`MonitorAppPermission`**`(java.lang.String name)`
Constructor for the MonitorAppPermission

# Method Summary

| Methods inherited from class java.security.BasicPermission |
|---|
| `equals, getActions, hashCode, implies, newPermissionCollection` |

| Methods inherited from class java.security.Permission |
|---|
| `checkGuard, getName, toString` |

| Methods inherited from class java.lang.Object |
|---|
| `clone, finalize, getClass, notify, notifyAll, wait, wait, wait` |

# Constructor Detail

### MonitorAppPermission

`public` **`MonitorAppPermission`**`(java.lang.String name)`
Constructor for the MonitorAppPermission
**Parameters:**
`name` - the name of this permission (see table in class description)

**org.ocap.system**

## Class RegisteredApiManager

```
java.lang.Object
  └ org.ocap.system.RegisteredApiManager
```

```
public abstract class RegisteredApiManager
extends java.lang.Object
```

This class represents a manager for registered APIs that can be registered with an implementation by a privileged application.

## Constructor Summary

| | |
|---|---|
| protected | **RegisteredApiManager**()<br>Protected constructor. |

## Method Summary

| | |
|---|---|
| static RegisteredApiManager | **getInstance**()<br>Gets the singleton instance of the Registered API manager. |
| abstract java.lang.String[] | **getNames**()<br>Gets a list of registered APIs. |
| abstract java.lang.String[] | **getUsedNames**()<br>Gets a list of registered APIs that are in use by the caller. |
| abstract java.lang.String | **getVersion**(java.lang.String name)<br>Gets the version of a registered API, or null if it is not registered. |
| abstract void | **register**(java.lang.String name, java.lang.String version, java.io.File scdf, short storagePriority)<br>Registers an API with the implementation. |
| abstract void | **unregister**(java.lang.String name)<br>Unregisters an API from the implementation. |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

## Constructor Detail

**RegisteredApiManager**

```
protected RegisteredApiManager()
```
Protected constructor.

# Method Detail

### getInstance

```
public static RegisteredApiManager getInstance()
```
Gets the singleton instance of the Registered API manager.

**Returns:**

The Registered API manager.

### register

```
public abstract void register(java.lang.String name,
                              java.lang.String version,
                              java.io.File scdf,
                              short storagePriority)
                       throws java.lang.IllegalArgumentException,
                              java.lang.IllegalStateException,
                              java.io.IOException
```
Registers an API with the implementation. If the name and version number matches an API already registered, this function does nothing (successfully). Matches for both name and version are based on exact case sensitive comparisons. If the name matches an API already registered, and the version number is different, the implementation SHALL: remove the existing API before installing the new API. The removal SHALL obey the semantics specified for the unregister() method. If the installation fails then the previously registered API SHALL NOT be removed. The removal of the previous API and installation of the new API SHALL be one atomic operation. (Note: This implies that the terminal MUST download and authenticate all files required for the new API, and only if this succeeds can it then remove the old API & install the new API. Application authors that do not need this behavior should note that unregistering the old API before registering a new version may reduce the memory usage of this operation and is strongly recommended). Paths in the SCDF are relative to the directory containing the SCDF. The priority field specified in the SCDF is ignored.

**Parameters:**

`name` - Name of the registered API.

`version` - Version of the registered API.

`scdf` - Path to the shared classes descriptor file.

`storagePriority` - Storage priority of classes in the SCDF.

**Throws:**

`java.lang.IllegalArgumentException` - if storagePriority is not a valid value as defined in chapter 12.

`java.lang.IllegalStateException` - if the API to be updated is in use by any application.

`java.io.IOException` - if the SCDF or any file listed in it does not exist, cannot be loaded, or are not correctly signed. Also thrown if the SCDF is not the correct format and cannot be parsed.

`java.lang.SecurityException` - if the calling application does not have MonitorAppPermission("registeredapi.manager").

### unregister

```
public abstract void unregister(java.lang.String name)
                         throws java.lang.IllegalArgumentException,
                                java.lang.IllegalStateException
```

Unregisters an API from the implementation. Removes all of the shared class files associated with the registered API from persistent and volatile memory. Removes the registered API from the registered API list.

**Parameters:**

`name` - Name of the registered API to unregister.

**Throws:**

`java.lang.IllegalArgumentException` - if no registered API with the name parameter has been registered.

`java.lang.IllegalStateException` - if the API to be unregistered is in use by any application.

`java.lang.SecurityException` - if the calling application does not have MonitorAppPermission("registeredapi.manager").

### getNames

```
public abstract java.lang.String[] getNames()
```

Gets a list of registered APIs. Note that this is intended for use by applications that manage registered APIs. Applications that use a registered API should call getUsedNames().

**Returns:**

An array of registered API names.

**Throws:**

`java.lang.SecurityException` - if the calling application does not have MonitorAppPermission("registeredapi.manager").

### getVersion

```
public abstract java.lang.String getVersion(java.lang.String name)
```

Gets the version of a registered API, or null if it is not registered.

**Parameters:**

`name` - the name of the registered API.

**Returns:**

the version of the registered API, or null if it is not registered.

**Throws:**

`java.lang.SecurityException` - if the calling application does not have MonitorAppPermission("registeredapi.manager") or RegisteredApiUserPermission(name).

### getUsedNames

```
public abstract java.lang.String[] getUsedNames()
```

Gets a list of registered APIs that are in use by the caller.

**Returns:**

An array of registered API names that are in use by the caller.

**org.ocap.system**

## Class RegisteredApiUserPermission

```
java.lang.Object
  └ java.security.Permission
      └ java.security.BasicPermission
          └ org.ocap.system.RegisteredApiUserPermission
```

**All Implemented Interfaces:**

java.io.Serializable, java.security.Guard

```
public final class RegisteredApiUserPermission
extends java.security.BasicPermission
```

The RegisteredApiUserPermission class represents permission for an application to use a specific registered API. When granted one of these permissions is created for each registered API the application is given access to based on a "registeredapi.user" entry in the application's PRF and where the name matches an ocap_j_registered_api_descriptor.

# Constructor Summary

| |
|---|
| **RegisteredApiUserPermission**(java.lang.String name)<br>Creates a new RegisteredApiUserPermission with the specified name. |
| **RegisteredApiUserPermission**(java.lang.String name, java.lang.String actions)<br>Creates a new RegisteredApiUserPermission with the specified name. |

# Method Summary

| **Methods inherited from class java.security.BasicPermission** |
|---|
| equals, getActions, hashCode, implies, newPermissionCollection |

| **Methods inherited from class java.security.Permission** |
|---|
| checkGuard, getName, toString |

| **Methods inherited from class java.lang.Object** |
|---|
| clone, finalize, getClass, notify, notifyAll, wait, wait, wait |

# Constructor Detail

**RegisteredApiUserPermission**

`public` **`RegisteredApiUserPermission`**`(java.lang.String name)`

Creates a new RegisteredApiUserPermission with the specified name.

**Parameters:**

`name` - The name of the registered API.

#### RegisteredApiUserPermission

`public` **`RegisteredApiUserPermission`**`(java.lang.String name, java.lang.String actions)`

Creates a new RegisteredApiUserPermission with the specified name.

**Parameters:**

`name` - The name of the registered API.

`actions` - This parameter is ignored.

**org.ocap.system**

## Interface SystemModule

```
public interface SystemModule
```

The SystemModule is used by an OCAP-J application that has a role of an assuming module to send an APDU to the CableCARD device. A SystemModule instance is provided by the `SystemModuleHandler.ready(org.ocap.system.SystemModule)` method after calling the `SystemModuleRegistrar.registerSASHandler(org.ocap.system.SystemModuleHandler, byte[])` method or the `SystemModuleRegistrar.registerMMIHandler(org.ocap.system.SystemModuleHandler)` method.

**See Also:**

`SystemModuleRegistrar`

# Method Summary

| | |
|---|---|
| void | **sendAPDU**(int apduTag, byte[] dataByte)<br>This method sends an APDU to the CableCARD device. |

# Method Detail

### sendAPDU

```
void sendAPDU(int apduTag,
              byte[] dataByte)
```

This method sends an APDU to the CableCARD device. The APDU structure is defined in Table 16 in Section 8.3 of EIA-679-B referred by CableCARD Interface 2.0 Specification [4] and SCTE 28 2003. The APDU structure consists of apdu_tag, length_field and data_byte.

For the Private Host Application of the SAS Resource, the session number for sending the APDU is decided by the OCAP implementation automatically when registration via the SystemModuleRegistrar.registerSASHandler(org.ocap.system.SystemModuleHandler, byte[]) method. Sending APDU is delegated to the SAS Resource.

For the MMI Resource and Application Information Resource, sending APDU is delegated to the resident MMI and Application Information Resources. The OCAP-J application can send APDUs of either MMI Resource or Application Information Resource via a single SystemModule. The OCAP implementation shall investigate the apdu_tag field in the APDU and send the APDU to the CableCARD device using the session of the Resource specified by the apdu_tag. The session established by the resident MMI and Application Information Resource are used to send the APDU.

For both above, the delegated Resource encodes the specified APDU into an SPDU complementing a length_field and sends it to the CableCARD device according to the OpenCable CableCARD Interface Specification.

The OCAP implementation doesn't have to confirm the validity of the specified dataByte parameter, but shall confirm the validity of the specified apduTag value.

This method returns immediately and doesn't confirm success of sending the APDU. Errors detected while sending the APDU are notified via the SystemModuleHandler.sendAPDUFailed(int, byte[]) method.

**Parameters:**

`apduTag` - an apdu_tag value for the APDU to be sent to the CableCARD device.

`dataByte` - a data_byte binary for the APDU to be sent to the CableCARD device. This value shall contain only the data_byte part of an APDU structure defined in the OpenCable CableCARD Interface Specification. The APDU consists of the specified apduTag and dataByte and a length_field complemented by the OCAP implementation.

**Throws:**

`java.lang.IllegalArgumentException` - if the specified apdu_tag value is invalid (i.e., the value is not among possible tag values for MMI Resource or Application Information Resource). Possible apdu_tag values and possible direction for each Resource are defined in the OpenCable CableCARD Interface Specification.

**org.ocap.system**

## Interface SystemModuleHandler

```
public interface SystemModuleHandler
```

The SystemModuleHandler is used by an OCAP-J application that has a role of an assuming module for the following purposes: 1) receive an APDU from the CableCARD device, 2) detect an unsent APDU to the POD in case of an error, and 3) notification of becoming registered and unregistered.

**See Also:**
`SystemModuleRegistrar`

# Method Summary

| | |
|---|---|
| void | **notifyUnregister**()<br>This is a call back method to notify that the SystemModuleHandler is being unregistered and give a chance to do a termination procedure. |
| void | **ready**(SystemModule systemModule)<br>This is a call back method to notify that this SystemModuleHandler is ready to receive an APDU, and returns a SystemModule to send an APDU to the CableCARD device. |
| void | **receiveAPDU**(int apduTag, int lengthField, byte[] dataByte)<br>This is a call back method to notify an APDU received from the CableCARD device. |
| void | **sendAPDUFailed**(int apduTag, byte[] dataByte)<br>This is a call back method to notify an error has occurred while sending an APDU via the `SystemModule.sendAPDU(int, byte[])` method. |

# Method Detail

**receiveAPDU**

```
void receiveAPDU(int apduTag,
                 int lengthField,
                 byte[] dataByte)
```

This is a call back method to notify an APDU received from the CableCARD device. The APDU structure is defined in Table 16 in Section 8.3 of EIA-679-B referred by CableCARD Interface 2.0 Specification [4] and SCTE 28 2003. The APDU structure consists of apdu_tag, length_field and data_byte.

For the Private Host Application on the SAS Resource, the SystemModuleHandler is bound to a specific session number (and a specific Private Host Application ID) when it is registered via the `SystemModuleRegistrar.registerSASHandler(org.ocap.system.SystemModuleHandler, byte[])` method. Only the receiveAPDU() method that is bound to the session of the received APDU shall be called only once by the OCAP implementation.

For the MMI Resource and the Application Information Resource, the OCAP-J application can receive APDUs for both Resources by a single SystemModuleHandler. The OCAP implementation shall call the receiveAPDU() method of the SystemModuleHandler registered via the `SystemModuleRegistrar.registerMMIHandler(org.ocap.system.SystemModuleHandler)` method only once for both the MMI and Application Information APDU.

The OCAP implementation extract the APDU from an SPDU from the CableCARD device according to the OpenCable CableCARD Interface Specification, and then call this method. Note that the OCAP implementation simply retrieves the field values from the APDU and call this method. No validity check is done by the OCAP implementation. Though SPDU and TPDU mechanism may detect a destruction of the APDU structure while transmitting, the OCAP shall call this method every time when it receive an APDU. In such case, the parameters may be invalid so that the OCAP-J application can detect an error.

Note that if the CableCARD device returns an APDU indicating an error condition, this method is called instead of the sendAPDUFailed() method.

This method shall return immediately.

**Parameters:**
`apduTag` - an apdu_tag value in the APDU coming from the CableCARD device. I.e., first 3 bytes. If the corresponding bytes are missed, they are filled by zero. Note that the OCAP implementation calls this method according to the session number, so the apdu_tag value may be out of the valid range.
`lengthField` - a length_field value in the APDU coming from the CableCARD device. This is a decimal int value converted from a length field encoded in ASN.1 BER. If the corresponding bytes are missing, the value of this parameter is set to 0.
`dataByte` - an data_byte bytes in the APDU coming from the CableCARD device. If the corresponding bytes are missed since signaling trouble, only existing bytes are specified. If they are more than expected length, all existing bytes are specified. The APDU consists of the specified apdu_tag, dataByte and length_field. The APDU format is defined in the CableCARD Interface 2.0 Specification .

### sendAPDUFailed

```
void sendAPDUFailed(int apduTag,
                    byte[] dataByte)
```

This is a call back method to notify an error has occurred while sending an APDU via the `SystemModule.sendAPDU(int, byte[])` method. This method shall return immediately.
**Parameters:**
`apduTag` - an apdu_tag of the APDU that was failed to be sent. This is the apduTag value specified in the SystemModule.sendAPDU() method.
`dataByte` - an data_byte of the APDU that was failed to be sent. This is dataByte value specified in the SystemModule.sendAPDU() method.

### notifyUnregister

```
void notifyUnregister()
```

This is a call back method to notify that the SystemModuleHandler is being unregistered and give a chance to do a termination procedure. This method returns after the termination procedure has finished.

### ready

```
void ready(SystemModule systemModule)
```

This is a call back method to notify that this SystemModuleHandler is ready to receive an APDU, and returns a SystemModule to send an APDU to the CableCARD device.
**Parameters:**
`systemModule` - a SystemModule instance corresponding to this SystemModuleHandler. The returned SystemModule sends an APDU using the same session that this SystemModuleHandler receives an APDU. Null is specified, if the OCAP implementation fails to establish a SAS connection or fails to create an SystemModule instance due to lack of resource.

**org.ocap.system**
## Class SystemModuleRegistrar

```
java.lang.Object
  └ org.ocap.system.SystemModuleRegistrar
```

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

This class is used by an OCAP-J application to assume the system modules. Each system module has a unique assuming mechanism.

**Private Host Application**
A Private Host Application on the Specific Application Support (SAS) Resource is an assumable system module. An OCAP-J application can register a SystemModuleHandler to take a role of a Private Host Application. If the SystemModuleHandler is registered successfully, the current Private Host Application that has a matching Private Host Application ID is terminated.

**Man Machine Interface (MMI) Resource and Application Information Resource**
The MMI Resource and the Application Information Resource are the assumable modules. But the assuming mechanism is different from the Private Host Application. These Resources assume the resident Resource at the same time. The resident MMI Resource and Application Information Resource don't terminate, but they shall pass all APDU to the assuming Resources. The resident MMI dialog can be hidden.

See also Section 20 *Baseline Functionality* for detailed assuming mechanism.

**Since:**
1.0

## Constructor Summary

| | |
|---|---|
| protected | **SystemModuleRegistrar**()<br>A constructor of this class. |

## Method Summary

| | |
|---|---|
| static SystemModuleRegistrar | **getInstance**()<br>This method returns a sole instance of the SystemModuleRegistrar class. |
| void | **registerMMIHandler**(SystemModuleHandler handler)<br>This method registers a SystemModuleHandler instance for the assuming MMI and Application Information Resource. |
| void | **registerSASHandler**(SystemModuleHandler handler, byte[] privateHostAppID)<br>This method registers the specified SystemModuleHandler instance for the specified privateHostAppID. |

# Method Summary

| | |
|---|---|
| void | **unregisterMMIHandler**()<br>This method unregisters the SystemModuleHandler and SystemModule instance of the assuming MMI and Application Information Resource and revives the resident MMI and Application Information Resource. |
| void | **unregisterSASHandler**(byte[] privateHostAppID)<br>This method unregisters the SystemModuleHandler and the SystemModule instance corresponding to the specified privateHostAppID, and revives an original resident Private Host Application. |
| void | **unregisterSASHandler**(SystemModuleHandler handler)<br>This method unregisters the specified SystemModuleHandler and the corresponding SystemModule instance, and revives an original resident Private Host Application. |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Constructor Detail

### SystemModuleRegistrar

protected **SystemModuleRegistrar**()

A constructor of this class. An application must use the getInstance() method to create an instance.

# Method Detail

### getInstance

public static SystemModuleRegistrar **getInstance**()

This method returns a sole instance of the SystemModuleRegistrar class. The SystemModuleRegistrar instance is either a singleton for each OCAP application or a singleton for an entire OCAP implementation.

**Returns:**

a singleton SystemModuleRegistrar instance.

**Throws:**

java.lang.SecurityException - if the caller does not have MonitorAppPermission("podApplication").

### registerSASHandler

public void **registerSASHandler**(SystemModuleHandler handler, byte[] privateHostAppID)

This method registers the specified SystemModuleHandler instance for the specified privateHostAppID. The Private Host Application is a logical entity defined in the CableCARD Interface 2.0 Specification.

If there is a current Private Host Application that has a matching Private Host Application ID as the privateHostAppID parameter, it shall be unregistered first, i.e., corresponding SystemModule and SystemModuleHandler shall be unregistered. The `SystemModuleHandler.notifyUnregister()` method of the SystemModuleHandler to be unregistered shall be called to notify its unregistration and give a chance to do a termination procedure. Note that the OCAP implementation shall call the notifyUnregister() method in a new thread to avoid blocking.

After the SystemModuleHandler.notifyUnregister() method returns, the OCAP implementation selects an appropriate session number for sending and receiving APDU. Then the OCAP implementation shall send the sas_connect_rqst APDU with the session automatically. After establishing the SAS connection, the OCAP implementation shall call the SystemModuleHandler.ready() method with a new SystemModule instance.

After ready() method is called, all APDUs shall be handled by the assuming OCAP-J application instead of the OCAP implementation.
If a native resident Private Host Application is implemented on the Host, it shall has Java interface and be registered in a same manner as the OCAP-J application. Only when no SystemModuleHandler that has a matching Private Host Application ID is registered by the Monitor Application, the OCAP implementation shall register such a native resident Private Host Application.
**Parameters:**
`handler` - a SystemModuleHandler instance to receive an APDU from the CableCARD device. If the handler has already been registered to the SystemModuleRegistrar, the method does nothing and throws IllegalArgumentException. Multiple call of this method with different SystemModuleHandler instance registers all of them.
`privateHostAppID` - a Private Host Application ID for the specified handler. This value is defined as an unsigned 64-bit value in the OpenCable CableCARD Interface specification. The specified byte array shall be big endian. This value is specified as the private_host_application_id field in the sas_connect_rqst APDU. If a SystemModuleHandler instance that has a matching privateHostAppID has already been registered, it shall be unregistered even if it is registered by another OCAP-J application.
**Throws:**
`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("podApplication").
`java.lang.IllegalStateException` - if the CableCARD device is not ready.
`java.lang.IllegalArgumentException` - if the specified handler already exists, or the specified parameter is out of range.

### unregisterSASHandler

`public void` **`unregisterSASHandler`**`(SystemModuleHandler handler)`

This method unregisters the specified SystemModuleHandler and the corresponding SystemModule instance, and revives an original resident Private Host Application.

In this method call, the `SystemModuleHandler.notifyUnregister()` method of the specified SystemModuleHandler shall be called to notify its unregistration and give a chance to do a termination procedure. The SystemModuleHandler and the corresponding SystemModule shall be removed from the SystemModuleRegistrar after returning of the notifyUnregister() method. Note that the OCAP implementation shall call the notifyUnregister() method in a new thread to avoid blocking.

The OCAP implementation shall re-register a native resident Private Host Application automatically (i.e., revive it), when no SystemModuleHandler that has a matching Private Host Application ID is registered.

**Parameters:**
`handler` - a SystemModuleHandler instance (the Private Host Application) to be unregistered. If the specified handler has not been registered to the SystemModuleRegistrar, the method call does nothing and throws IllegalArgumentException.
**Throws:**

`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("podApplication")
`java.lang.IllegalArgumentException` - if the specified handler has not been registered or if either parameter is null.

### unregisterSASHandler

```
public void unregisterSASHandler(byte[] privateHostAppID)
```

This method unregisters the SystemModuleHandler and the SystemModule instance corresponding to the specified privateHostAppID, and revives an original resident Private Host Application.

In this method call, the `SystemModuleHandler.notifyUnregister()` method corresponding to the specified privateHostAppID shall be called to notify its unregistration and give a chance to do a termination procedure. The SystemModuleHandler and the corresponding SystemModule shall be removed from the SystemModuleRegistrar after returning of the notifyUnregister() method. Note that the OCAP implementation shall call the notifyUnregister() method in a new thread to avoid blocking.

The OCAP implementation shall re-register a native resident Private Host Application automatically (i.e., revive it), when no SystemModuleHandler that has a matching Private Host Application ID is registered.

**Parameters:**
`privateHostAppID` - a Private Host Application ID of the Private Host Application (i.e., SystemModuleHandler) to be unregistered. This value is defined as an unsigned 64-bit value in the OpenCable Host-POD Interface specification. The specified byte array shall be a big endian. If the specified privateHostAppID has not been registered to the SystemModuleRegistrar, this method does nothing and throws IllegalArgumentException.
**Throws:**
`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("podApplication").
`java.lang.IllegalArgumentException` - if the specified privateHostAppID has not been registered.

### registerMMIHandler

```
public void registerMMIHandler(SystemModuleHandler handler)
```

This method registers a SystemModuleHandler instance for the assuming MMI and Application Information Resource. The OCAP implementation shall call the `SystemModuleHandler.ready(org.ocap.system.SystemModule)` method with a new SystemModule instance to send an APDU to the CableCARD device.

The resident MMI and Application Information Resources don't terminate but shall pass APDU to the SystemModuleHandler. The OCAP-J application can send and receive APDUs via the `SystemModule.sendAPDU(int, byte[])` and the `SystemModuleHandler.receiveAPDU(int, int, byte[])` method instead of the resident Resources. The sessions established by the resident MMI and Application Information Resource is used to send and receive the APDU. See also the description of the `SystemModule` and the `SystemModuleHandler`.

After successful registration, the resident MMI Resource shall not represent the MMI dialog on the screen. The Host shall close all resident MMI dialog and finalize all transaction related to the MMI dialog. The Host shall send the close_mmi_cnf APDU to the CableCARD device to notify MMI dialog closing. If the unregisterMMIHandler() is called or the OCAP-J application that called this method changes its state to Destroyed, the resident MMI Resource can represent the MMI dialog again.

**Parameters:**

`handler` - a SystemModuleHandler instance to receive an APDU from CableCARD device. Only one SystemModuleHandler can be registered. If the second SystemModuleHandler is to be registered, this method throws IllegalArgumentException.

**Throws:**

`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("podApplication").

`java.lang.IllegalStateException` - if the CableCARD device is not ready.

`java.lang.IllegalArgumentException` - if the second SystemModuleHandler is to be registered.

### unregisterMMIHandler

`public void` **`unregisterMMIHandler`**`()`

This method unregisters the SystemModuleHandler and SystemModule instance of the assuming MMI and Application Information Resource and revives the resident MMI and Application Information Resource.

In this method call, the `SystemModuleHandler.notifyUnregister()` method of the SystemModuleHandler registered by the registerMMIHandler() method shall be called to notify its unregistration and give a chance to do a termination procedure. At least, all MMI dialog shall be closed and all of the transaction related to the MMI and Application Information Resource shall be terminated. The OCAP-J application shall send the close_mmi_cnf APDU to the CableCARD device to notify MMI dialog closing. The SystemModuleHandler and the corresponding SystemModule shall be removed from the SystemModuleRegistrar after returning of the notifyUnregister() method. I.e., after returning of the notifyUnregister() method, no APDU can be sent. Note that the OCAP implementation shall call the notifyUnregister() method in a new thread to avoid blocking.

After this method is called, the resident MMI and Application Information Resource handles all APDUs and the resident MMI can represent the MMI dialog again.

**Throws:**

`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("podApplication")

# Annex R OCAP 1.1 Hardware POD API

This section presents the `org.ocap.hardware.pod` APIs.

***Table R–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| Annex R OCAP 1.1 Hardware POD API | No Corresponding Section | OCAP-Specific Extension |

## Package org.ocap.hardware.pod

This package provides a way to set and get OpenCable CableCARD Resource related parameters.

| Interface Summary | |
|---|---|
| **HostParamHandler** | A class that implements this interface can reject the update of the Feature parameter in the Host device. |
| **PODApplication** | This class represents an Application that resides in the OpenCable CableCARD device. |

| Class Summary | |
|---|---|
| **POD** | This class provides an access to functions and information of the OpenCable CableCARD device on the OCAP Host device. |

## Package org.ocap.hardware.pod Description

This package provides a way to set and get OpenCable CableCARD Resource related parameters.

The following CableCARD Resources are covered by this package:

- Application Information Resource
- Generic Feature Control Support Resource

The `POD` class provides access to information and functions related to the CableCARD Resources listed above. It is based on a generic singleton model; only privileged applications that have MonitorAppPermission("podApplication"), such as the Monitor Application, can get an instance of it via the POD.getInstance() method. Other `POD` methods do not check the permissions settings of the invoking object.

### Application Information Resource

The CableCARD device has zero or more `PODApplication`s. The `POD.getApplications()` method returns an array of `PODApplication` instances. The `PODApplication` class provides access to the parameters defined in the Application Information Resource of the OpenCable CableCARD Interface Specification.

Example:

```
import org.ocap.hardware.pod.*;
```

```
...
POD pod = POD.getInstance();
PODApplication[] apps = pod.getApplications();
String name = apps[0].getName();
...
```

#### Generic Feature Control Support Resource

OCAP applications may modify the Feature parameter in the Host device via the `POD.updateHostParam(int, byte[])` method. Applications can also get notified and reject update of the Feature parameter via the `HostParamHandler`, when the CableCARD device attempts to change Feature parameters. The `HostParamHandler` is registered via the `POD.setHostParamHandler(org.ocap.hardware.pod.HostParamHandler)` method.

Example:

```
import org.ocap.hardware.pod.*;
...
POD pod = POD.getInstance();
int acOutlet = 7;
byte[] unswitched = {0X02};
byte[] value = pod.getHostParam(acOutlet);
...
pod.updateHostParam(acOutlet, unswitched);
...
```

### org.ocap.hardware.pod
## Interface HostParamHandler

```
public interface HostParamHandler
```

A class that implements this interface can reject the update of the Feature parameter in the Host device. Feature parameter is defined for the Generic Feature Control Support in the OpenCable CableCARD Interface specification. An OCAP-J application can set only one instance of such classes to the OCAP implementation via the `POD.setHostParamHandler(org.ocap.hardware.pod.HostParamHandler)` method.

Before Feature parameter in the Host is modified, the `notifyUpdate(int, byte[])` method shall be called with the Feature ID to be modified and its Feature parameter value. And only if the HostParamHandler.notifyUpdate() method returns true, the Feature parameter value in the Host device will be modified. Note that the Host device may reject the update of Feature parameter even if the HostParamHandler.notifyUpdate() method returns true.

The Feature ID and the Feature parameter value format are defined in the CableCARD Interface 2.0 Specification [4]. For example, the Feature ID of "RF Output Channel" Feature is 0x1, and its parameter value format is

```
Rf_output_channel() {
    Output_channel
    Output_channel_ui
}
```

The Feature parameters in the Host device will be modified by the following cases.

- The CableCARD sends feature_parameters APDU to the Host. (See [CCIF 2.0].)
- The Host modifies its own Feature parameters.

- An OCAP-J application calls the `POD.updateHostParam(int, byte[])` method.

In every cases, the HostParamHandler.notifyUpdate() method shall be called.

# Method Summary

| | |
|---|---|
| `boolean` | **`notifyUpdate`**`(int featureID, byte[] value)`<br>This is a call back method to notify an update of the Feature parameter in the Host device. |

# Method Detail

### notifyUpdate

```
boolean notifyUpdate(int featureID,
                     byte[] value)
```

This is a call back method to notify an update of the Feature parameter in the Host device. This method shall be called every time before the Feature parameter is modified. Only if this method returns true, the Host device can modify its Feature parameter by the specified value.

Note that the Host device may reject the update of Feature parameter even if the HostParamHandler.notifyUpdate() method returns true.

This method should return immediately without blocking.

**Parameters:**
`featureID` - a Feature ID for the Generic Feature Control Support in the CableCARD Interface 2.0 Specification [4]. The Feature ID reserved for proprietary use (0x70 - 0xFF) can be specified.
`value` - a Feature parameter value for the specified featureID. An actual format of each Feature parameter is defined in the CableCARD Interface 2.0 Specification [4]. For example, if the featureID is 0x1, the value is

```
Rf_output_channel() {
    Output_channel
    Output_channel_ui
}
```

**Returns:**
`true` to accept the modification of the specified value. `false` to reject it.
**See Also:**
`POD.setHostParamHandler(org.ocap.hardware.pod.HostParamHandler)`

**org.ocap.hardware.pod**

## Class POD

```
java.lang.Object
  └ org.ocap.hardware.pod.POD

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

This class provides an access to functions and information of the OpenCable CableCARD device on the OCAP Host device. The following functions and information are provided.

- Get a list of all applications in the CableCARD device.
- Get Feature list supported by the Host.
- Get a manufacture ID and a version number of the CableCARD device.
- Get a current status of the CableCARD device.
- Update the Feature parameter in the Host.
- Reject updating of the Feature parameter in the Host.

## Constructor Summary

| | |
|---|---|
| protected | **POD**()<br>A constructor of this class. |

## Method Summary

| | |
|---|---|
| PODApplication[] | **getApplications**()<br>This method returns the CableCARD device applications listed in the Application_info_cnf() APDU defined in the OpenCable CableCARD Interface specification. |
| int[] | **getHostFeatureList**()<br>This method returns a list of the Feature IDs supported by the Host device. |
| byte[] | **getHostParam**(int featureID)<br>This method returns the current Feature parameter value in the Host device for the specified Feature ID. |
| static POD | **getInstance**()<br>This method returns the sole instance of the POD class. |
| int | **getManufacturerID**()<br>This method returns a CableCARD device manufacturer ID. |
| int | **getVersionNumber**()<br>This method returns a CableCARD device version number. |

# Method Summary

| | |
|---|---|
| boolean | **isReady**()<br>This method provides a current status of the CableCARD device. |
| void | **setHostParamHandler**(HostParamHandler handler)<br>This method sets an instance of a class that implements the HostParamHandler interface. |
| boolean | **updateHostParam**(int featureID, byte[] value)<br>This method updates the Feature parameter value in the Host device. |

| **Methods inherited from class java.lang.Object** |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Constructor Detail

### POD

protected **POD**()

A constructor of this class. An application must use the `getInstance()` method to create an instance.

# Method Detail

### getInstance

public static POD **getInstance**()

This method returns the sole instance of the POD class. The POD instance is either a singleton for each OCAP application or a singleton for an entire OCAP implementation.

**Returns:**

a singleton POD instance.

**Throws:**

`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("podApplication").

### isReady

public boolean **isReady**()

This method provides a current status of the CableCARD device.

**Returns:**

true if the CableCARD device has completed the booting process.

### getManufacturerID

public int **getManufacturerID**()

This method returns a CableCARD device manufacturer ID.

**Returns:**

a pod_manufacturer_id in the Application_info_cnf() APDU defined in the CableCARD Interface 2.0 Specification [4].

**Throws:**

`java.lang.IllegalStateException` - if the CableCARD is not ready, i.e., the `isReady()` method returns false.

#### getVersionNumber

```
public int getVersionNumber()
```

This method returns a CableCARD device version number.

**Returns:**

pod_version_number in the Application_info_cnf() APDU defined in the CableCARD Interface 2.0 Specification [4].

**Throws:**

`java.lang.IllegalStateException` - if the CableCARD is not ready, i.e., the `isReady()` method returns false.

#### getApplications

```
public PODApplication[] getApplications()
```

This method returns the CableCARD device applications listed in the Application_info_cnf() APDU defined in the OpenCable CableCARD Interface specification.

Note that the Host need not to send the Application_info_req APDU. It may cache the information.

**Returns:**

a list of CableCARD device applications in the CableCARD device.

**Throws:**

`java.lang.IllegalStateException` - if the CableCARD is not ready, i.e., the `isReady()` method returns false.

#### getHostFeatureList

```
public int[] getHostFeatureList()
```

This method returns a list of the Feature IDs supported by the Host device. Feature ID is defined in the OpenCable CableCARD Interface specification.

**Returns:**

a list of Feature IDs supported by the Host device.

#### updateHostParam

```
public boolean updateHostParam(int featureID,
                               byte[] value)
```

This method updates the Feature parameter value in the Host device. In this method call, the `HostParamHandler.notifyUpdate(int, byte[])` method shall be called. The notifyUpdate() method may reject update of the Feature parameter and also the Host device may reject it. The updated Feature parameter shall be notified to the CableCARD device according to the CableCARD Interface 2.0 Specification [4] after this method returns, but this method doesn't confirm a successful notification to the CableCARD device.

The Feature ID and Feature parameter format is defined in the [CCIF 2.0]. See also the `HostParamHandler` for more information.

Note that the `HostParamHandler.notifyUpdate(int, byte[])` method shall be called before the Feature parameter is updated by this method call.

**Parameters:**
`featureID` - a Feature ID to be updated. Feature ID is defined in the CableCARD Interface 2.0 Specification [4]. The Feature ID reserved for proprietary use (0x70 - 0xFF) can be specified.
`value` - a new Feature parameter value for the specified featureID. An actual format of each Feature parameter is defined in the CableCARD Interface 2.0 Specification [4]. For example, if the featureID is 0x1, the value is

```
Rf_output_channel() {
    Output_channel
    Output_channel_ui
}
```

**Returns:**
`true` if update was successful. `false` if rejected by the Host.
**Throws:**
`java.lang.IllegalArgumentException` - if the specified featureID is not in a range of 0 <= featureID <= 0xFF, or the value is null.
**See Also:**
`HostParamHandler`

### getHostParam

`public byte[] **getHostParam**(int featureID)`

This method returns the current Feature parameter value in the Host device for the specified Feature ID. The Feature ID and Feature parameter format is defined in the CableCARD Interface 2.0 Specification [4]. See also the `HostParamHandler` for more information.
**Parameters:**
`featureID` - a Feature ID defined in the CableCARD Interface 2.0 Specification [4]. The Feature ID reserved for proprietary use (0x70 - 0xFF) can be specified.
**Returns:**
a current Feature parameter value for the specified featureID. For example, if the featureID is 0x1, the value is

```
Rf_output_channel() {
    Output_channel
    Output_channel_ui
}
```

An array of length zero, if the specified featureID is not supported.
**Throws:**
`java.lang.IllegalArgumentException` - if the specified featureID is not in a range of 0 <= featureID <= 0xFF.
**See Also:**
`HostParamHandler`

### setHostParamHandler

`public void **setHostParamHandler**(HostParamHandler handler)`

This method sets an instance of a class that implements the HostParamHandler interface. Only one instance of such class can be set to the OCAP system. Multiple calls of this method replace the previous instance by a new one. By default, no HostParamHandler is set, i.e., all update of Feature parameter is decided by the Host device.
**Parameters:**
`handler` - an instance of a class that implements the HostParamHandler. if `null` is specified, the current HostParamHandler is removed.

**See Also:**
`HostParamHandler`

**org.ocap.hardware.pod**

## Interface PODApplication

```
public interface PODApplication
```

This class represents an Application that resides in the OpenCable CableCARD device. The CableCARD device has zero or more CableCARD Applications. PODApplication instances corresponding to those CableCARD Applications are retrieved by the `POD.getApplications()` method. This class provides information of the CableCARD Application described in the Application_info_req() APDU defined in the OpenCable CableCARD Interface specification [4].

## Field Summary

| | |
|---|---|
| `static int` | **`TYPE_CA`**<br>The Conditional Access application type. |
| `static int` | **`TYPE_CP`**<br>The "Copy Protection" application type. |
| `static int` | **`TYPE_DIAGNOSTIC`**<br>The "Diagnostic" application type. |
| `static int` | **`TYPE_DVS167`**<br>The "Network Interface - DVS/167" application type. |
| `static int` | **`TYPE_DVS178`**<br>The "Network Interface - DVS/178" application type. |
| `static int` | **`TYPE_IP`**<br>The "IP Service" application type. |
| `static int` | **`TYPE_UNDESIGNATED`**<br>The "Undesignated" application type. |

## Method Summary

| | |
|---|---|
| `java.lang.String` | **`getName()`**<br>This method returns an application name of the CableCARD Application represented by this class. |
| `int` | **`getType()`**<br>This method returns an application type value of the CableCARD Application represented by this class. |
| `java.lang.String` | **`getURL()`**<br>This method returns a URL of the CableCARD Application represented by this class. |
| `int` | **`getVersionNumber()`**<br>This method returns an application version number of the CableCARD Application represented by this class. |

# Field Detail

### TYPE_CA

`static final int` **`TYPE_CA`**

The Conditional Access application type. This value is defined for the application_type field in the Application_info_cnf() APDU. See [CCIF 2.0].

### TYPE_CP

`static final int` **`TYPE_CP`**

The "Copy Protection" application type. This value is defined for the application_type field in the Application_info_cnf() APDU. See [CCIF 2.0].

### TYPE_IP

`static final int` **`TYPE_IP`**

The "IP Service" application type. This value is defined for the application_type field in the Application_info_cnf() APDU. See [CCIF 2.0].

### TYPE_DVS167

`static final int` **`TYPE_DVS167`**

The "Network Interface - DVS/167" application type. This value is defined for the application_type field in the Application_info_cnf() APDU. See [CCIF 2.0].

### TYPE_DVS178

`static final int` **`TYPE_DVS178`**

The "Network Interface - DVS/178" application type. This value is defined for the application_type field in the Application_info_cnf() APDU. See [CCIF 2.0].

### TYPE_DIAGNOSTIC

`static final int` **`TYPE_DIAGNOSTIC`**

The "Diagnostic" application type. This value is defined for the application_type field in the Application_info_cnf() APDU. See [CCIF 2.0].

### TYPE_UNDESIGNATED

`static final int` **`TYPE_UNDESIGNATED`**

The "Undesignated" application type. This value is defined for the application_type field in the Application_info_cnf() APDU. See [CCIF 2.0].

# Method Detail

### getType

`int` **`getType`**`()`

This method returns an application type value of the CableCARD Application represented by this class. The application type is described in the application_type field in the Application_info_cnf() APDU.

**Returns:**

an application type value of the CableCARD application represented by this class. Known values are defined as the field values prefixed with "TYPE_".

### getVersionNumber

`int` **`getVersionNumber`**`()`

This method returns an application version number of the CableCARD Application represented by this class. The application version number is described in the application_version_number field in the Application_info_cnf() APDU.

**Returns:**

an application version number value of the CableCARD Application represented by this class.

### getName

`java.lang.String` **`getName`**`()`

This method returns an application name of the CableCARD Application represented by this class. The application version number is described in the application_name_byte field in the Application_info_cnf() APDU.

**Returns:**

an application name of the CableCARD Application represented by this class.

### getURL

`java.lang.String` **`getURL`**`()`

This method returns a URL of the CableCARD Application represented by this class. The URL is described in the application_url_byte field in the Application_info_cnf() APDU.

**Returns:**

a URL of the CableCARD Application represented by this class.

**Throws:**

`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("podApplication").

# Annex S OCAP 1.1 Media API

This section presents the `org.ocap.media` APIs.

***Table S–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| Annex S OCAP 1.1 Media API | No Corresponding Section | OCAP-Specific Extension |

## Package org.ocap.media

| Interface Summary | |
|---|---|
| **AlternativeMediaPresentationReason** | This interface represents possible reasons that lead to alternative media presentation. |
| **AnalogAudioChangeListener** | This interface is to receive a notification when secondary analog audio is newly appeared or disappeared. |
| **AnalogAudioControl** | This interface is used to get current analog audio types and select a specific analog audio type in a running JMF player. |
| **ClosedCaptioningControl** | This interface is used to turn closed-captioning in a running JMF player on and off and to select a captioning service (C1 to C4 and T1 to T4) to be represented. |
| **ClosedCaptioningListener** | This is a listener interface to receive a notification when the state of closed-captioning has changed or a new closed-captioning service (C1 to C4, T1 to T4) is selected. |
| **MediaAccessAuthorization** | A `MediaAccessAuthorization` object represents the presentation authorization given by a registered MediaAccessHandler for a specific A/V content. |
| **MediaAccessConditionControl** | This interface allows an application to notify that conditions of media presentation in a running JMF player have been modified, and so the check for media presentation must be done. |
| **MediaAccessHandler** | A class implementing this interface can prevent the presentation of A/V service components. |
| **MediaTimerListener** | This is a listener to inform events on a MediaTimer object. |
| **NotPresentedMediaInterface** | NotPresentedMediaInterface shall be implemented by classes which can report failure to access media components. |
| **VBIFilter** | This class represents a VBI filter. |
| **VBIFilterListener** | This interface represents a VBI filter event listener. |

| Class Summary | |
|---|---|
| **AlternativeMediaPresentationEvent** | `AlternativeMediaPresentationEvent` is a JMF event generated to indicate that an "alternative" content is presented during the media presentation of a service. |

| Class Summary | |
|---|---|
| **AnalogAudioChangeEvent** | |
| **ClosedCaptioningAttribute** | This class represents a system wide preference of closed-captioning representation. |
| **ClosedCaptioningEvent** | This class is an event to notify a change of a closed-captioning state. |
| **FilterResourceAvailableEvent** | This event notifies an application that a VBIFilterGroup released VBIFilters, i.e., another application may have an opportunity to reserve new VBIFilters. |
| **ForcedDisconnectedEvent** | This event indicates a VBIFilterGroup is detached from a ServiceContext for any reason. |
| **MediaAccessHandlerRegistrar** | This class allows to register a handler that can prevent the presentation of A/V service components. |
| **MediaPresentationEvaluationTrigger** | This class represents possible reasons to trigger an evaluation that leads to the generation of an `AlternativeMediaPresentationEvent` or a `NormalMediaPresentationEvent`. |
| **MediaPresentationEvent** | `MediaPresentationEvent` is a JMF event used as the parent class of events indicating dynamic changes to the presentation of media components. |
| **MediaTimer** | This is a timer class that counts time based on a media time of a specified Player. |
| **NormalMediaPresentationEvent** | `NormalMediaPresentationEvent` is a JMF event generated when the normal media components of a service are presented. |
| **VBIFilterEvent** | This class represents a VBI filter event. |
| **VBIFilterGroup** | This class represents a group of VBI data filters. |

### org.ocap.media
## Class AlternativeMediaPresentationEvent

```
java.lang.Object
  └ java.util.EventObject
      └ javax.media.ControllerEvent
          └ javax.media.TransitionEvent
              └ org.ocap.media.MediaPresentationEvent
                  └ org.ocap.media.AlternativeMediaPresentationEvent
```

**All Implemented Interfaces:**

java.io.Serializable, javax.media.MediaEvent, NotPresentedMediaInterface

```
public abstract class AlternativeMediaPresentationEvent
extends MediaPresentationEvent
implements NotPresentedMediaInterface
```

`AlternativeMediaPresentationEvent` is a JMF event generated to indicate that an "alternative" content is presented during the media presentation of a service.

Alternative content is defined as content that is not actually part of the service.

`AlternativeMediaPresentationEvent` notification is generated :

- When alternative media content presentation begins;
- During the presentation of a service, if any of the service components presented are replaced by alternative content;
- During the presentation of a service, if an alternative media content was presented and an evaluation leads to a new alternative media content presentation.

# Field Summary

| Fields inherited from class java.util.EventObject |
|---|
| source |

# Constructor Summary

| | |
|---|---|
| protected | **AlternativeMediaPresentationEvent**(javax.media.Controller from, int previous, int current, int target)<br>Constructor of MediaPresentationEvent |

# Method Summary

| | |
|---|---|
| org.davic.mpeg.ElementaryStream[] | **getNotPresentedStreams**() |
| int | **getReason**(org.davic.mpeg.ElementaryStreams) |

| Methods inherited from class org.ocap.media.MediaPresentationEvent |
|---|
| getPresentedStreams, getSourceURL, getTrigger, isSourceDigital |

| Methods inherited from class javax.media.TransitionEvent |
|---|
| getCurrentState, getPreviousState, getTargetState, toString |

| Methods inherited from class javax.media.ControllerEvent |
|---|
| getSource, getSourceController |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

# Constructor Detail

**AlternativeMediaPresentationEvent**

```
protected AlternativeMediaPresentationEvent(javax.media.Controller from,
                                            int previous,
                                            int current,
                                            int target)
```

Constructor of MediaPresentationEvent

**See Also:**
`MediaPresentationEvent`

# Method Detail

**getNotPresentedStreams**

```
public org.davic.mpeg.ElementaryStream[] getNotPresentedStreams()
```

**Specified by:**
`getNotPresentedStreams` in interface `NotPresentedMediaInterface`

**Returns:**
Returns the subset of explicitly (by Application request) or implicitly (by the Player itself) service components that were selected and which presentation was not possible.

**getReason**

```
public int getReason(org.davic.mpeg.ElementaryStream es)
```

**Specified by:**
`getReason` in interface `NotPresentedMediaInterface`

**Parameters:**
`es` - a not presented service component.

**Returns:**
Returns a bit mask of reasons that lead to the non presentation of the given service component. The reasons are defined in `AlternativeMediaPresentationReason`)interface.

**org.ocap.media**

## Interface AlternativeMediaPresentationReason

```
public interface AlternativeMediaPresentationReason
```

This interface represents possible reasons that lead to alternative media presentation. Registered `MediaAccessHandler` can define its own reason and pass them to the OCAP implementation through the MediaAccessHandler.checkMediaAccessAuthorization() method.

# Field Summary

| | |
|---|---|
| `static int` | **`BROADCAST_INCONSISTENCY`**<br>Bit indicating that broadcast information is inconsistent : for example PMT is missing. |
| `static int` | **`CA_UNKNOWN`**<br>Bit indicating that media are ciphered and the CA does not correspond to ciphering. |
| `static int` | **`COMMERCIAL_DIALOG`**<br>Bit indicating that a user dialog for payment is necessary before media presentation. |
| `static int` | **`HARDWARE_RESOURCE_NOT_AVAILABLE`**<br>Bit indicating that hardware resource necessary for presenting service components is not available. |
| `static int` | **`NO_ENTITLEMENT`**<br>Bit indicating that service components are ciphered and the user has no entitlement to view all or part of them. |
| `static int` | **`RATING_PROBLEM`**<br>Reason indicating that media presentation is not authorized regarding to the program rating. |
| `static int` | **`REASON_FIRST`**<br>Marks the first bit for the range of alternative media presentation reasons. |
| `static int` | **`REASON_LAST`**<br>Marks the last bit for the range of alternative media presentation reasons. |

# Field Detail

### REASON_FIRST

```
static final int REASON_FIRST
```

Marks the first bit for the range of alternative media presentation reasons.

### NO_ENTITLEMENT

```
static final int NO_ENTITLEMENT
```

Bit indicating that service components are ciphered and the user has no entitlement to view all or part of them.

#### COMMERCIAL_DIALOG

`static final int` **`COMMERCIAL_DIALOG`**

Bit indicating that a user dialog for payment is necessary before media presentation.

#### RATING_PROBLEM

`static final int` **`RATING_PROBLEM`**

Reason indicating that media presentation is not authorized regarding to the program rating.

#### CA_UNKNOWN

`static final int` **`CA_UNKNOWN`**

Bit indicating that media are ciphered and the CA does not correspond to ciphering.

#### BROADCAST_INCONSISTENCY

`static final int` **`BROADCAST_INCONSISTENCY`**

Bit indicating that broadcast information is inconsistent : for example PMT is missing.

#### HARDWARE_RESOURCE_NOT_AVAILABLE

`static final int` **`HARDWARE_RESOURCE_NOT_AVAILABLE`**

Bit indicating that hardware resource necessary for presenting service components is not available.

#### REASON_LAST

`static final int` **`REASON_LAST`**

Marks the last bit for the range of alternative media presentation reasons.

**org.ocap.media**

## Class AnalogAudioChangeEvent

```
java.lang.Object
  └ java.util.EventObject
      └ org.ocap.media.AnalogAudioChangeEvent
```

**All Implemented Interfaces:**

java.io.Serializable

```
public class AnalogAudioChangeEvent
extends java.util.EventObject
```

## Field Summary

| | |
|---|---|
| static int | **EVENTID_SECONDARY_AUDIO_AVAILABLE**<br>This event indicates a secondary audio is newly found in the currently playing analog service. |
| static int | **EVENTID_SECONDARY_AUDIO_UNAVAILABLE**<br>This event indicates a secondary audio is disappeared in the currently playing analog service. |

| Fields inherited from class java.util.EventObject |
|---|
| source |

## Constructor Summary

| |
|---|
| **AnalogAudioChangeEvent**(java.lang.Object source, int id)<br>Constructor of a new AnalogAudioChangeEvent with the specified event ID. |

## Method Summary

| | |
|---|---|
| int | **getEventID**()<br>Get the event ID associated with this AnalogAudioChangeEvent. |

| Methods inherited from class java.util.EventObject |
|---|
| getSource, toString |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

# Field Detail

### EVENTID_SECONDARY_AUDIO_AVAILABLE

```
public static final int EVENTID_SECONDARY_AUDIO_AVAILABLE
```

This event indicates a secondary audio is newly found in the currently playing analog service.

### EVENTID_SECONDARY_AUDIO_UNAVAILABLE

```
public static final int EVENTID_SECONDARY_AUDIO_UNAVAILABLE
```

This event indicates a secondary audio is disappeared in the currently playing analog service.

# Constructor Detail

### AnalogAudioChangeEvent

```
public AnalogAudioChangeEvent(java.lang.Object source,
                              int id)
```

Constructor of a new AnalogAudioChangeEvent with the specified event ID.

**Parameters:**

`source` - The Controller that generates this event.

`id` - The event ID. One of EVENTID_SECONDARY_AUDIO_AVAILABLE and EVENTID_SECONDARY_AUDIO_UNAVAILABLE.

# Method Detail

### getEventID

```
public int getEventID()
```

Get the event ID associated with this AnalogAudioChangeEvent.

**Returns:**

an event ID of this AnalogAudioChangeEvent.

**org.ocap.media**

## Interface AnalogAudioChangeListener

**All Superinterfaces:**
java.util.EventListener

```
public interface AnalogAudioChangeListener
extends java.util.EventListener
```

This interface is to receive a notification when secondary analog audio is newly appeared or disappeared.

# Method Summary

| | |
|---|---|
| void | **analogAudioChanged**(AnalogAudioChangeEvent event)<br>This method is called when secondary analog audio is newly detected or disappeared. |

# Method Detail

**analogAudioChanged**

```
void analogAudioChanged(AnalogAudioChangeEvent event)
```

This method is called when secondary analog audio is newly detected or disappeared.

**Parameters:**
`event` - AnalogAudioChangeEvent

**org.ocap.media**
## Interface AnalogAudioControl

**All Superinterfaces:**
javax.media.Control

```
public interface AnalogAudioControl
extends javax.media.Control
```

This interface is used to get current analog audio types and select a specific analog audio type in a running JMF player. Instance of the AnalogaudioControl interface shall be obtained via `Controller.getControl(java.lang.String)` and a `Controller.getControls()` methods by all applications. But MonitorAppPermission("handler.analogAudioControl") is necessary to call methods in this interface

# Field Summary

| | |
|---|---|
| `static int` | **`ANALOG_NOT_PLAYING`**<br>No analog audio is playing. |
| `static int` | **`ANALOG_PRIMARY_AUDIO`**<br>Indicates an primary analog audio. |
| `static int` | **`ANALOG_SECONDARY_AUDIO`**<br>Indicates an secondary analog audio. |

# Method Summary

| | |
|---|---|
| `void` | **`addAnalogAudioChangeListener`**`(AnalogAudioChangeListener listener)`<br>Adds a listener to notify a change of secondary audio existence. |
| `int` | **`getCurrentAudioType`**`()`<br>Returns the currently playing analog audio type. |
| `boolean` | **`hasSecondaryAudio`**`()`<br>Check if the secondary audio exists. |
| `void` | **`removeAnalogAudioChangeListener`**`(AnalogAudioChangeListener listener)`<br>Removes the specified AnalogAudioChangeListener. |
| `void` | **`selectAudio`**`(int audioType)`<br>Changes currently playing audio type. |

| **Methods inherited from interface javax.media.Control** |
|---|
| `getControlComponent` |

# Field Detail

### ANALOG_PRIMARY_AUDIO

```
static final int ANALOG_PRIMARY_AUDIO
```
Indicates an primary analog audio.

### ANALOG_SECONDARY_AUDIO

```
static final int ANALOG_SECONDARY_AUDIO
```
Indicates an secondary analog audio.

### ANALOG_NOT_PLAYING

```
static final int ANALOG_NOT_PLAYING
```
No analog audio is playing.

# Method Detail

### addAnalogAudioChangeListener

```
void addAnalogAudioChangeListener(AnalogAudioChangeListener listener)
                                  throws java.lang.SecurityException
```
Adds a listener to notify a change of secondary audio existence. Multiple calls with same listener is simply ignored with throwing no exception.

**Parameters:**

`listener` - a AnalogAudioChangeListener instance to notify a change of secondary audio existence.

**Throws:**

`java.lang.SecurityException` - if a caller does not have MonitorAppPermission("handler.analogAudioControl")

### removeAnalogAudioChangeListener

```
void removeAnalogAudioChangeListener(AnalogAudioChangeListener listener)
                                     throws java.lang.SecurityException
```
Removes the specified AnalogAudioChangeListener. This method does nothing if the specified listener is not previously added or already removed.

**Parameters:**

`listener` - a AnalogAudioChangeListener instance to be removed.

**Throws:**

`java.lang.SecurityException` - if a caller does not have MonitorAppPermission("handler.analogAudioControl")

### getCurrentAudioType

```
int getCurrentAudioType()
                         throws java.lang.SecurityException
```
Returns the currently playing analog audio type. If there is no selected analog audio, -1 is returned.

**Returns:**

Currently selected analog audio type. One of ANALOG_PRIMARY_AUDIO, ANALOG_SECONDARY_AUDIO and ANALOG_NOT_PLAYING (If there is no selected audio).

**Throws:**

`java.lang.SecurityException` - if a caller does not have MonitorAppPermission("handler.analogAudioControl")

### hasSecondaryAudio

```
boolean hasSecondaryAudio()
                          throws java.lang.SecurityException
```

Check if the secondary audio exists.

**Returns:**

true if the secondary audio exists.

**Throws:**

`java.lang.SecurityException` - if a caller does not have MonitorAppPermission("handler.analogAudioControl")

### selectAudio

```
void selectAudio(int audioType)
                 throws java.lang.SecurityException
```

Changes currently playing audio type.

**Parameters:**

`audioType` - Audio type to be played. One of ANALOG_PRIMARY_AUDIO and ANALOG_SECONDARY_AUDIO.

**Throws:**

`java.lang.SecurityException` - if a caller does not have MonitorAppPermission("handler.analogAudioControl")

**org.ocap.media**

## Class ClosedCaptioningAttribute

```
java.lang.Object
  └ org.ocap.media.ClosedCaptioningAttribute
```

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

This class represents a system wide preference of closed-captioning representation. The OCAP implementation shall display closed-captioning according to preference values that is specified by this class. Application developers should be aware that the FCC has defined strict rules regarding display of CC and EAS (see http://ftp.fcc.gov/cgb/dro/caption.html for FCC closed captioning rules).

# Field Summary

| | |
|---|---|
| static int | **CC_ATTRIBUTE_FONT_ITALICIZED**<br>Indicates a font face attribute of a closed-captioning text. |
| static int | **CC_ATTRIBUTE_FONT_STYLE**<br>Indicates a font style attribute of a closed-captioning text. |
| static int | **CC_ATTRIBUTE_FONT_UNDERLINE**<br>Indicates a font face attribute of a closed-captioning text. |
| static int | **CC_ATTRIBUTE_PEN_BG_COLOR**<br>Indicates a pen back ground color attribute to draw closed-captioning text. |
| static int | **CC_ATTRIBUTE_PEN_BG_OPACITY**<br>Indicates a pen back ground opacity attribute of a closed-captioning text. |
| static int | **CC_ATTRIBUTE_PEN_FG_COLOR**<br>Indicates a pen color attribute to draw closed-captioning text. |
| static int | **CC_ATTRIBUTE_PEN_FG_OPACITY**<br>Indicates a pen opacity attribute of a closed-captioning text. |
| static int | **CC_ATTRIBUTE_PEN_SIZE**<br>Indicates a font size attribute of a closed-captioning text. |
| static int | **CC_ATTRIBUTE_WINDOW_BORDER_COLOR**<br>Indicates a border color attribute of a closed-captioning window. |
| static int | **CC_ATTRIBUTE_WINDOW_BORDER_TYPE**<br>Indicates a border type attribute of a closed-captioning window. |
| static int | **CC_ATTRIBUTE_WINDOW_FILL_COLOR**<br>Indicates a window fill color attribute of a closed-captioning window. |
| static int | **CC_ATTRIBUTE_WINDOW_FILL_OPACITY**<br>Indicates a border type attribute of a closed-captioning window. |
| static int | **CC_BORDER_DEPRESSED**<br>Indicates a border type of DEPRESSED. |
| static int | **CC_BORDER_NONE**<br>Indicates a border type of NONE. |

## Field Summary

| | |
|---|---|
| static int | **CC_BORDER_RAISED**<br>Indicates a border type of RAISED. |
| static int | **CC_BORDER_SHADOW_LEFT**<br>Indicates a border type of SHADOW_LEFT. |
| static int | **CC_BORDER_SHADOW_RIGHT**<br>Indicates a border type of SHADOW_RIGHT. |
| static int | **CC_BORDER_UNIFORM**<br>Indicates a border type of UNIFORM. |
| static int | **CC_OPACITY_FLASH**<br>Indicates a opacity value for a flash. |
| static int | **CC_OPACITY_SOLID**<br>Indicates a opacity value for a solid. |
| static int | **CC_OPACITY_TRANSLUCENT**<br>Indicates a opacity value for a translucent. |
| static int | **CC_OPACITY_TRANSPARENT**<br>Indicates a opacity value for a transparent. |
| static int | **CC_PEN_SIZE_LARGE**<br>Indicates a large pen size. |
| static int | **CC_PEN_SIZE_SMALL**<br>Indicates a small pen size. |
| static int | **CC_PEN_SIZE_STANDARD**<br>Indicates a standard pen size. |
| static int | **CC_TYPE_ANALOG**<br>Indicates an analog type closed-captioning. |
| static int | **CC_TYPE_DIGITAL**<br>Indicates an digital type closed-captioning. |

## Constructor Summary

| | |
|---|---|
| protected | **ClosedCaptioningAttribute**()<br>A constructor of this class. |

## Method Summary

| | |
|---|---|
| java.lang.Object | **getCCAttribute**(int attribute, int ccType)<br>This method returns a current attribute values applied to a closed-captioning text on a screen. |
| java.lang.Object[] | **getCCCapability**(int attribute, int ccType)<br>This method returns a possible attribute values applied to an closed-captioning text on a screen. |

# Method Summary

| | |
|---|---|
| `static ClosedCaptioningAttribute` | **`getInstance`**`()`<br>This method returns an instance of this class. |
| `void` | **`setCCAttribute`**`(int[] attribute, java.lang.Object[] value, int ccType)`<br>This method sets a preferred attribute values applied to a closed-captioning text on a screen. |

| **Methods inherited from class java.lang.Object** |
|---|
| `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait` |

# Field Detail

### CC_ATTRIBUTE_PEN_FG_COLOR

`public static final int` **`CC_ATTRIBUTE_PEN_FG_COLOR`**

Indicates a pen color attribute to draw closed-captioning text. Identical to the "fg color" parameter of SetPenColor command of EIA-708-B. For an analog captioning, an equivalent attribute is assigned.

### CC_ATTRIBUTE_PEN_BG_COLOR

`public static final int` **`CC_ATTRIBUTE_PEN_BG_COLOR`**

Indicates a pen back ground color attribute to draw closed-captioning text. Identical to the "bg color" parameter of SetPenColor command of EIA-708-B. For an analog captioning, an equivalent attribute is assigned.

### CC_ATTRIBUTE_PEN_FG_OPACITY

`public static final int` **`CC_ATTRIBUTE_PEN_FG_OPACITY`**

Indicates a pen opacity attribute of a closed-captioning text. Identical to the "fg opacity" parameter of SetPenColor command of EIA-708-B. For an analog captioning, an equivalent attribute is assigned.

### CC_ATTRIBUTE_PEN_BG_OPACITY

`public static final int` **`CC_ATTRIBUTE_PEN_BG_OPACITY`**

Indicates a pen back ground opacity attribute of a closed-captioning text. Identical to the "bg opacity" parameter of SetPenColor command of EIA-708-B. For an analog captioning, an equivalent attribute is assigned.

### CC_ATTRIBUTE_FONT_STYLE

`public static final int` **`CC_ATTRIBUTE_FONT_STYLE`**

Indicates a font style attribute of a closed-captioning text. Identical to the "font style" parameter of SetPenAttributes command of EIA-708-B. For an analog captioning, an equivalent attribute is assigned.

#### CC_ATTRIBUTE_PEN_SIZE

`public static final int` **`CC_ATTRIBUTE_PEN_SIZE`**

Indicates a font size attribute of a closed-captioning text. Identical to the "pen size" parameter of SetPenAttributes command of EIA-708-B. For an analog captioning, an equivalent attribute is assigned.

#### CC_ATTRIBUTE_FONT_ITALICIZED

`public static final int` **`CC_ATTRIBUTE_FONT_ITALICIZED`**

Indicates a font face attribute of a closed-captioning text. Identical to the "italics" parameter of SetPenAttributes command of EIA-708-B. For an analog captioning, an equivalent attribute is assigned.

#### CC_ATTRIBUTE_FONT_UNDERLINE

`public static final int` **`CC_ATTRIBUTE_FONT_UNDERLINE`**

Indicates a font face attribute of a closed-captioning text. Identical to the "underline" parameter of SetPenAttributes command of EIA-708-B. For an analog captioning, an equivalent attribute is assigned.

#### CC_ATTRIBUTE_WINDOW_FILL_COLOR

`public static final int` **`CC_ATTRIBUTE_WINDOW_FILL_COLOR`**

Indicates a window fill color attribute of a closed-captioning window. Identical to the "fill color" parameter of SetWindowAttributes command of EIA-708-B. For an analog captioning, an equivalent attribute is assigned.

#### CC_ATTRIBUTE_WINDOW_FILL_OPACITY

`public static final int` **`CC_ATTRIBUTE_WINDOW_FILL_OPACITY`**

Indicates a border type attribute of a closed-captioning window. Identical to the "fill opacity" parameter of SetWindowAttributes command of EIA-708-B. For an analog captioning, an equivalent attribute is assigned.

#### CC_ATTRIBUTE_WINDOW_BORDER_TYPE

`public static final int` **`CC_ATTRIBUTE_WINDOW_BORDER_TYPE`**

Indicates a border type attribute of a closed-captioning window. Identical to the "border color" parameter of SetWindowAttributes command of EIA-708-B. For an analog captioning, an equivalent attribute is assigned.

#### CC_ATTRIBUTE_WINDOW_BORDER_COLOR

`public static final int` **`CC_ATTRIBUTE_WINDOW_BORDER_COLOR`**

Indicates a border color attribute of a closed-captioning window. Identical to the "border color" parameter of SetWindowAttributes command of EIA-708-B. For an analog captioning, an equivalent attribute is assigned.

#### CC_PEN_SIZE_SMALL

`public static final int` **`CC_PEN_SIZE_SMALL`**
Indicates a small pen size.

#### CC_PEN_SIZE_STANDARD

`public static final int` **`CC_PEN_SIZE_STANDARD`**
Indicates a standard pen size.

#### CC_PEN_SIZE_LARGE

`public static final int` **`CC_PEN_SIZE_LARGE`**
Indicates a large pen size.

#### CC_OPACITY_SOLID

`public static final int` **`CC_OPACITY_SOLID`**
Indicates a opacity value for a solid.

#### CC_OPACITY_FLASH

`public static final int` **`CC_OPACITY_FLASH`**
Indicates a opacity value for a flash.

#### CC_OPACITY_TRANSLUCENT

`public static final int` **`CC_OPACITY_TRANSLUCENT`**
Indicates a opacity value for a translucent.

#### CC_OPACITY_TRANSPARENT

`public static final int` **`CC_OPACITY_TRANSPARENT`**
Indicates a opacity value for a transparent.

#### CC_BORDER_NONE

`public static final int` **`CC_BORDER_NONE`**
Indicates a border type of NONE.

#### CC_BORDER_RAISED

`public static final int` **`CC_BORDER_RAISED`**
Indicates a border type of RAISED.

#### CC_BORDER_DEPRESSED

```
public static final int CC_BORDER_DEPRESSED
```
Indicates a border type of DEPRESSED.

#### CC_BORDER_UNIFORM

```
public static final int CC_BORDER_UNIFORM
```
Indicates a border type of UNIFORM.

#### CC_BORDER_SHADOW_LEFT

```
public static final int CC_BORDER_SHADOW_LEFT
```
Indicates a border type of SHADOW_LEFT.

#### CC_BORDER_SHADOW_RIGHT

```
public static final int CC_BORDER_SHADOW_RIGHT
```
Indicates a border type of SHADOW_RIGHT.

#### CC_TYPE_ANALOG

```
public static final int CC_TYPE_ANALOG
```
Indicates an analog type closed-captioning.

#### CC_TYPE_DIGITAL

```
public static final int CC_TYPE_DIGITAL
```
Indicates an digital type closed-captioning.

## Constructor Detail

#### ClosedCaptioningAttribute

```
protected ClosedCaptioningAttribute()
```
A constructor of this class. An application shall not call this constructor directly.

## Method Detail

#### getInstance

```
public static ClosedCaptioningAttribute getInstance()
```
This method returns an instance of this class. It is not required to be a singleton manner.

**Returns:**

A ClosedCaptioningAttribute instance.

**Throws:**

`java.lang.SecurityException` - if the caller doesn't have MonitorAppPermission("handler.closedCaptioning").

### getCCCapability

```
public java.lang.Object[] getCCCapability(int attribute,
                                          int ccType)
```

This method returns a possible attribute values applied to an closed-captioning text on a screen. Note that the possible font attribute may be different from the possible font for Java application since the closed-captioning module may be implemented by native language.

**Parameters:**
`attribute` - specify an attribute to get possible values. One of constants that has a CC_ATTRIBUTE_ prefix shall be specified.
`ccType` - either CC_ANALOG or CC_DIGITAL to specify a type of closed-captioning.
**Returns:**
an array of possible attribute values of an closed-captioning text corresponding to the specified attribute parameter.

- If the attribute parameter is CC_ATTRIBUTE_PEN_FG_COLOR or CC_ATTRIBUTE_PEN_BG_COLOR, an array of java.awt.Color that represents possible font color returns. The Color.getString() shall return a text expression of its color to show a user.
- If the attribute parameter is CC_ATTRIBUTE_PEN_FG_OPACITY or CC_ATTRIBUTE_PEN_BG_OPACITY, an array of constants that represents possible opacity returns. The opacity constants has a prefix of CC_OPACITY_.
- If the attribute parameter is CC_ATTRIBUTE_FONT_STYLE, an array of String that represents possible font style returns. It is recommended that the String is one of font style defined in EIA-708-B but not restricted to it. The host device can provide a new style.
- If the attribute parameter is CC_ATTRIBUTE_PEN_SIZE, an array of constants that represents possible pen size returns. The pen size constants has a prefix of CC_PEN_SIZE_.
- If the attribute parameter is CC_ATTRIBUTE_FONT_ITALICIZED, an array of possible Integer value (YES=1, NO=0) returns. I.e., if the host can select a plane font or an italicized font, an array of [0, 1] (or [1, 0]) returns. If the host only supports a plane font, [0] returns.
- If the attribute parameter is CC_ATTRIBUTE_FONT_UNDERLINE, an array of possible Integer value (YES=1, NO=0) returns. See also the CC_ATTRIBUTE_FONT_ITALICIZED description.
- If the attribute parameter is CC_ATTRIBUTE_WINDOW_FILL_COLOR, an array of java.awt.Color that represents possible window fill color returns. The Color.getString() shall return a text expression of its color to show a user.
- If the attribute parameter is CC_ATTRIBUTE_WINDOW_FILL_OPACITY an array of constants that represents possible opacity returns. The opacity constants has a prefix of CC_OPACITY_.
- If the attribute parameter is CC_ATTRIBUTE_WINDOW_BORDER_TYPE an array of constants that represents possible border type returns. The border type constants has a prefix of CC_BORDER_.
- If the attribute parameter is CC_ATTRIBUTE_WINDOW_BORDER_COLOR, an array of java.awt.Color that represents possible window border color returns. The Color.getString() shall return a text expression of its color to show a user.

**Throws:**
`java.lang.IllegalArgumentException` - if a specified attribute or ccType parameter is out of range.

### setCCAttribute

```
public void setCCAttribute(int[] attribute,
                           java.lang.Object[] value,
                           int ccType)
```

This method sets a preferred attribute values applied to a closed-captioning text on a screen. Some attribute values can be specified by one call of this method. If one of the specified attribute value is invalid, i.e., the value is not included in the return value of the `getCCCapability(int, int)` method, this method changes none of current attribute values and throw an exception.

**Parameters:**
`attribute` - an array of attributes to be set a preferred value. One of constants that has a CC_ATTRIBUTE_ prefix shall be specified.
`value` - an array of preferred values to be used to draw a closed-captioning text. The value shall be one of the return value from the `getCCCapability(int, int)` method for the specified attribute, or null to set a host's default value. The i-th item of the value array corresponds to the i-th item of the attribute array.
`ccType` - either CC_ANALOG or CC_DIGITAL to specify a type of closed-captioning.
**Throws:**
`java.lang.IllegalArgumentException` - if a specified attribute, value, or ccType parameter is out of range or not a capable value, or if a length of a specified attribute array doesn't matches with a length of a specified value array.

### getCCAttribute

```
public java.lang.Object getCCAttribute(int attribute,
                                       int ccType)
```

This method returns a current attribute values applied to a closed-captioning text on a screen.

**Parameters:**
`attribute` - specify an attribute to get a preferred values. One of constants that has a CC_ATTRIBUTE_ prefix shall be specified. See the `getCCCapability(int, int)` method also.
`ccType` - either CC_ANALOG or CC_DIGITAL to specify a type of closed-captioning.
**Returns:**
a current attribute value corresponding to the specified closed-captioning attribute parameter. See the `getCCCapability(int, int)` method for an applicable value.
**Throws:**
`java.lang.IllegalArgumentException` - if a specified attribute or ccType parameter is out of range.

**org.ocap.media**

## Interface ClosedCaptioningControl

**All Superinterfaces:**
javax.media.Control

```
public interface ClosedCaptioningControl
extends javax.media.Control
```

This interface is used to turn closed-captioning in a running JMF player on and off and to select a captioning service (C1 to C4 and T1 to T4) to be represented. Instance of the ClosedCaptioningControl interface shall be obtained via a `Controller.getControl(java.lang.String)` and a `Controller.getControls()` method by all applications. But MonitorAppPermission("handler.closedCaptioning") is necessary to call methods in this interface.

The captioning text is represented according to preferred attribute values set by a `ClosedCaptioningAttribute` class.

# Field Summary

| | |
|---|---|
| static int | **CC_ANALOG_SERVICE_CC1**<br>Indicates an analog closed-captioning service CC1. |
| static int | **CC_ANALOG_SERVICE_CC2**<br>Indicates an analog closed-captioning service CC2. |
| static int | **CC_ANALOG_SERVICE_CC3**<br>Indicates an analog closed-captioning service CC3. |
| static int | **CC_ANALOG_SERVICE_CC4**<br>Indicates an analog closed-captioning service CC4. |
| static int | **CC_ANALOG_SERVICE_T1**<br>Indicates an analog closed-captioning service T1. |
| static int | **CC_ANALOG_SERVICE_T2**<br>Indicates an analog closed-captioning service T2. |
| static int | **CC_ANALOG_SERVICE_T3**<br>Indicates an analog closed-captioning service T3. |
| static int | **CC_ANALOG_SERVICE_T4**<br>Indicates an analog closed-captioning service T4. |
| static int | **CC_TURN_OFF**<br>Indicates turn digital/analog closed-captioning on. |
| static int | **CC_TURN_ON**<br>Indicates turn digital/analog closed-captioning off. |
| static int | **CC_TURN_ON_MUTE**<br>Indicates turn digital/analog closed-captioning on only when muting an audio. |

# Method Summary

| | |
|---|---|
| void | **addClosedCaptioningListener**(ClosedCaptioningListener ccListener)<br>Add a listener to notify a closed-captioning state change. |
| int[] | **getClosedCaptioningServiceNumber**()<br>This method returns a current closed-captioning service for a JMF Player that is controlled by a ClosedCaptioningControl instance. |
| int | **getClosedCaptioningState**()<br>Get the current state of closed-captioning of a JMF Player that is controlled by a ClosedCaptioningControl instance. |
| int[] | **getSupportedClosedCaptioningServiceNumber**()<br>This method returns closed-captioning service numbers that are supported by a JMF Player that is controlled by a ClosedCaptioningControl instance. |
| void | **removeClosedCaptioningListener**(ClosedCaptioningListener ccListener)<br>Remove the given ClosedCaptioningListener. |
| void | **setClosedCaptioningServiceNumber**(int analogServiceNumber, int digitalServiceNumber)<br>This method sets a new closed-captioning service number to be represented by a JMF Player that is controlled by a ClosedCaptioningControl instance. |
| void | **setClosedCaptioningState**(int turnOn)<br>Turn closed-captioning of a JMF Player that is controlled by a ClosedCaptioningControl instance on or off. |

| **Methods inherited from interface javax.media.Control** |
|---|
| getControlComponent |

# Field Detail

### CC_ANALOG_SERVICE_CC1

static final int **CC_ANALOG_SERVICE_CC1**
Indicates an analog closed-captioning service CC1.

### CC_ANALOG_SERVICE_CC2

static final int **CC_ANALOG_SERVICE_CC2**
Indicates an analog closed-captioning service CC2.

### CC_ANALOG_SERVICE_CC3

static final int **CC_ANALOG_SERVICE_CC3**
Indicates an analog closed-captioning service CC3.

#### CC_ANALOG_SERVICE_CC4

`static final int` **`CC_ANALOG_SERVICE_CC4`**
Indicates an analog closed-captioning service CC4.

#### CC_ANALOG_SERVICE_T1

`static final int` **`CC_ANALOG_SERVICE_T1`**
Indicates an analog closed-captioning service T1.

#### CC_ANALOG_SERVICE_T2

`static final int` **`CC_ANALOG_SERVICE_T2`**
Indicates an analog closed-captioning service T2.

#### CC_ANALOG_SERVICE_T3

`static final int` **`CC_ANALOG_SERVICE_T3`**
Indicates an analog closed-captioning service T3.

#### CC_ANALOG_SERVICE_T4

`static final int` **`CC_ANALOG_SERVICE_T4`**
Indicates an analog closed-captioning service T4.

#### CC_TURN_OFF

`static final int` **`CC_TURN_OFF`**
Indicates turn digital/analog closed-captioning on.

#### CC_TURN_ON

`static final int` **`CC_TURN_ON`**
Indicates turn digital/analog closed-captioning off.

#### CC_TURN_ON_MUTE

`static final int` **`CC_TURN_ON_MUTE`**
Indicates turn digital/analog closed-captioning on only when muting an audio.

# Method Detail

#### setClosedCaptioningState

`void` **`setClosedCaptioningState`**`(int turnOn)`
Turn closed-captioning of a JMF Player that is controlled by a ClosedCaptioningControl instance on or off. Note that only one closed-captioning decoding may be supported on the OCAP implementation at once.

This method may turn off closed-captioning of another JMF Player automatically. Such a automatic turn off is notified by a `ClosedCaptioningEvent` event.

**Parameters:**

`turnOn` - An integer value specifying whether to turn closed-captioning on, off or "on mute". CC_TURN_ON, to turn closed-captioning represented by a `Player` instance that is controlled by the ClosedCaptioningControl instance on. CC_TURN_OFF, to turn it off. CC_TURN_ON_MUTE, to turn it on only when muting an audio.

**Throws:**

`java.lang.SecurityException` - if the caller doesn't have MonitorAppPermission("handler.closedCaptioning").

### getClosedCaptioningState

```
int getClosedCaptioningState()
```

Get the current state of closed-captioning of a JMF Player that is controlled by a ClosedCaptioningControl instance. Note that this method returns a current status set by the setClosedCaptioningState() method or automatic turning off brought by the method call. This method doesn't care if an actual caption channel packet in the MPEG video header or line 21 data exist, or if an Caption Service Descriptor has changed.

**Returns:**

An integer value representing a closed captioning state. One of CC TURN ON, CC TURN OFF, and CC TURN ON MUTE.

**Throws:**

`java.lang.SecurityException` - if the caller doesn't have MonitorAppPermission("handler.closedCaptioning").

### setClosedCaptioningServiceNumber

```
void setClosedCaptioningServiceNumber(int analogServiceNumber,
                                      int digitalServiceNumber)
```

This method sets a new closed-captioning service number to be represented by a JMF Player that is controlled by a ClosedCaptioningControl instance.

Captioning text will be rendered when captioning is turned on and corresponding captioning text data is transmitted. When an analog video is played on the JMF player, captioning service of an analogServiceNumber in VBI signal defined by EIA-608-B analog closed captioning will be rendered. When a digital video is played on the JMF player, captioning service of a digitalServiceNumber in MPEG picture header defined by EIA-708-B digital closed captioning will be rendered. If MPEG picture header doesn’t contain a captioning service of the digitalServiceNumber, a captioning service of analogServiceNumber in the MPEG picture header may be used instead.

The previously represented caption service shall be disappeared, when a new service is set. This method doesn't check if the specified closed-captioning service is transmitted with the current video actually.

**Parameters:**

`analogServiceNumber` - An integer representing an analog closed-captioning service number. The serviceNumber value shall be a return value of the `getSupportedClosedCaptioningServiceNumber()` method and shall be an analog captioning service, i.e., have a CC_ANALOG_SERVICE prefix. "-1" if no decoding of analog captioning is necessary.

`digitalServiceNumber` - An integer representing a digital closed-captioning service number. The serviceNumber value shall be a return value of the `getSupportedClosedCaptioningServiceNumber()` method and shall be an digital captioning service. "-1" if no decoding of digital captioning is necessary.

**Throws:**
`java.lang.IllegalArgumentException` - if the serviceNumber is not a return value of the getSupportedClosedCaptioningServiceNumber() method.
`java.lang.SecurityException` - if the caller doesn't have MonitorAppPermission("handler.closedCaptioning").

### getClosedCaptioningServiceNumber

`int[]` **`getClosedCaptioningServiceNumber`**`()`

This method returns a current closed-captioning service for a JMF Player that is controlled by a ClosedCaptioningControl instance. This method doesn't care if the specified closed-captioning service is transmitted in the current video.

**Returns:**
An array of integers representing a closed-captioning services. The first item shall be an analog captioning service number and the second item shall be an digital captioning service number. The array length shall be 2. A value "-1" indicates no captioning service number is specified for the captioning type.

**Throws:**
`java.lang.SecurityException` - if the caller doesn't have MonitorAppPermission("handler.closedCaptioning").

### getSupportedClosedCaptioningServiceNumber

`int[]` **`getSupportedClosedCaptioningServiceNumber`**`()`

This method returns closed-captioning service numbers that are supported by a JMF Player that is controlled by a ClosedCaptioningControl instance. Only service number returned by this method can be specified to the `setClosedCaptioningServiceNumber(int, int)` method. Note that this method doesn't check if the returned closed-captioning service is transmitted with the current video actually. I.e., this method returns just a capability of the host device.

**Returns:**
An array of closed-captioning service number that are supported by a JMF Player that is controlled by a ClosedCaptioningControl instance. If the service is analog captioning of EIA-608-B, the returned service number value shall be one of constants that has a prefix of CC_ANALOG_SERVICE_. If the service is digital captioning of EIA-708-B, the returned service number value shall be an actual service number (1 to 63).

**Throws:**
`java.lang.SecurityException` - if the caller doesn't have MonitorAppPermission("handler.closedCaptioning").

### addClosedCaptioningListener

`void` **`addClosedCaptioningListener`**`(ClosedCaptioningListener ccListener)`

Add a listener to notify a closed-captioning state change. Multiple calls with same ccListener instance is simply ignored with throwing no exception.

**Parameters:**
`ccListener` - a `ClosedCaptioningListener` instance to notify a change of the closed-captioning state.

**Throws:**
`java.lang.SecurityException` - if the caller doesn't have MonitorAppPermission("handler.closedCaptioning").

#### removeClosedCaptioningListener

`void` **`removeClosedCaptioningListener`**`(ClosedCaptioningListener ccListener)`

Remove the given ClosedCaptioningListener. This method does nothing if the specified ccListener is null, not previously added or already removed.

**Parameters:**

`ccListener` - a `ClosedCaptioningListener` instance to be removed.

**Throws:**

`java.lang.SecurityException` - if the caller doesn't have MonitorAppPermission("handler.closedCaptioning").

**org.ocap.media**

## Class ClosedCaptioningEvent

```
java.lang.Object
  └ java.util.EventObject
      └ org.ocap.media.ClosedCaptioningEvent
```

**All Implemented Interfaces:**

java.io.Serializable

```
public class ClosedCaptioningEvent
extends java.util.EventObject
```

This class is an event to notify a change of a closed-captioning state.

# Field Summary

| | |
|---|---|
| static int | **EVENTID_CLOSED_CAPTIONING_OFF**<br>This event indicates a current closed-captioning state is turning off. |
| static int | **EVENTID_CLOSED_CAPTIONING_ON**<br>This event indicates a current closed-captioning state is turning on. |
| static int | **EVENTID_CLOSED_CAPTIONING_ON_MUTE**<br>This event indicates a current closed-captioning state is "on mute". |
| static int | **EVENTID_CLOSED_CAPTIONING_SELECT_NEW_SERVICE**<br>This event indicates a new closed-captioning service (C1 to C4, T1 to T4) is selected. |

| Fields inherited from class java.util.EventObject |
|---|
| source |

# Constructor Summary

| |
|---|
| **ClosedCaptioningEvent**(java.lang.Object source, int id)<br>Construct a new ClosedCaptioningEvent with the specified event ID. |

# Method Summary

| | |
|---|---|
| int | **getEventID**()<br>Get the event ID associated with this ClosedCaptioningEvent. |

| Methods inherited from class java.util.EventObject |
|---|
| getSource, toString |

| Methods inherited from class java.lang.Object |
| --- |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

# Field Detail

### EVENTID_CLOSED_CAPTIONING_ON

```
public static final int EVENTID_CLOSED_CAPTIONING_ON
```

This event indicates a current closed-captioning state is turning on.

### EVENTID_CLOSED_CAPTIONING_OFF

```
public static final int EVENTID_CLOSED_CAPTIONING_OFF
```

This event indicates a current closed-captioning state is turning off.

### EVENTID_CLOSED_CAPTIONING_ON_MUTE

```
public static final int EVENTID_CLOSED_CAPTIONING_ON_MUTE
```

This event indicates a current closed-captioning state is "on mute".

### EVENTID_CLOSED_CAPTIONING_SELECT_NEW_SERVICE

```
public static final int EVENTID_CLOSED_CAPTIONING_SELECT_NEW_SERVICE
```

This event indicates a new closed-captioning service (C1 to C4, T1 to T4) is selected.

# Constructor Detail

### ClosedCaptioningEvent

```
public ClosedCaptioningEvent(java.lang.Object source,
                             int id)
```

Construct a new ClosedCaptioningEvent with the specified event ID.

**Parameters:**

`source` - The object where the event originated.

`id` - The event ID. One of the following values: EVENTID_CLOSED_CAPTIONING_ON, EVENTID_CLOSED_CAPTIONING_OFF, EVENTID_CLOSED_CAPTIONING_ON_MUTE and EVENTID_CLOSED_CAPTIONING_SELECT_NEW_SERVICE.

# Method Detail

### getEventID

```
public int getEventID()
```

Get the event ID associated with this ClosedCaptioningEvent.

**Returns:**

An integer representation of the event ID.

**org.ocap.media**

## Interface ClosedCaptioningListener

**All Superinterfaces:**

java.util.EventListener

```
public interface ClosedCaptioningListener
extends java.util.EventListener
```

This is a listener interface to receive a notification when the state of closed-captioning has changed or a new closed-captioning service (C1 to C4, T1 to T4) is selected.

A listener instance is added to a ClosedCaptioningControl instance via the `ClosedCaptioningControl.addClosedCaptioningListener(org.ocap.media.ClosedCaptioningListener)` method.

**Version:**

1.0

**Author:**

Mark S. Millard (Vidiom Systems), Shigeaki Watanabe (Panasonic): modify

# Method Summary

| | |
|---|---|
| void | **ccStatusChanged**(ClosedCaptioningEvent event)<br>This method shall be called when a closed-captioning state or a captioning service of the JMF Player that is controled by the ClosedCaptioningControl instance has changed. |

# Method Detail

### ccStatusChanged

```
void ccStatusChanged(ClosedCaptioningEvent event)
```

This method shall be called when a closed-captioning state or a captioning service of the JMF Player that is controled by the ClosedCaptioningControl instance has changed. Note that this method is not called when an existance of an actual caption channel packet in the MPEG video header or line 21 data has changed. This method is not called when an Caption Service Descriptor has changed.

**Parameters:**

`event` - The closed-captioning status event.

**org.ocap.media**

## Class FilterResourceAvailableEvent

```
java.lang.Object
  └ java.util.EventObject
      └ org.davic.resources.ResourceStatusEvent
          └ org.ocap.media.FilterResourceAvailableEvent
```

**All Implemented Interfaces:**

java.io.Serializable

```
public class FilterResourceAvailableEvent
extends org.davic.resources.ResourceStatusEvent
```

This event notifies an application that a VBIFilterGroup released VBIFilters, i.e., another application may have an opportunity to reserve new VBIFilters. This event is just a hint of resource status, so that the filters may not be available when an application calls a `VBIFilterGroup.attach(javax.tv.service.selection.ServiceContext, org.davic.resources.ResourceClient, java.lang.Object)` method actually.

# Field Summary

| Fields inherited from class java.util.EventObject |
|---|
| source |

# Constructor Summary

| |
|---|
| **FilterResourceAvailableEvent**(VBIFilterGroup f)<br>A constructor of this class. |

# Method Summary

| | |
|---|---|
| java.lang.Object | **getSource**()<br>This method returns an instance of a class implementing VBIFilterGroup that is the source of the event. |

| Methods inherited from class java.util.EventObject |
|---|
| toString |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

# Constructor Detail

### FilterResourceAvailableEvent

```
public FilterResourceAvailableEvent(VBIFilterGroup f)
```

A constructor of this class.

**Parameters:**

`f` - Instance of a VBIFilterGroup that is the source of this event.

# Method Detail

### getSource

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

This method returns an instance of a class implementing VBIFilterGroup that is the source of the event.

**Overrides:**

`getSource` in class `org.davic.resources.ResourceStatusEvent`

**Returns:**

instance of a class implementing VBIFilterGroup that is the source of the event

**org.ocap.media**

## Class ForcedDisconnectedEvent

```
java.lang.Object
  └ java.util.EventObject
      └ org.davic.resources.ResourceStatusEvent
          └ org.ocap.media.ForcedDisconnectedEvent
```

**All Implemented Interfaces:**

java.io.Serializable

```
public class ForcedDisconnectedEvent
extends org.davic.resources.ResourceStatusEvent
```

This event indicates a VBIFilterGroup is detached from a ServiceContext for any reason. A `ResourceClient.notifyRelease(org.davic.resources.ResourceProxy)` is also called to inform all filters held by the VBIFilterGroup have been forcibly released.

# Field Summary

| Fields inherited from class java.util.EventObject |
|---|
| source |

# Constructor Summary

| |
|---|
| **ForcedDisconnectedEvent**(VBIFilterGroup f)<br>A constructor of this class. |

# Method Summary

| | |
|---|---|
| java.lang.Object | **getSource**()<br>This method returns an instance of a class implementing VBIFilterGroup that is the source of the event. |

| Methods inherited from class java.util.EventObject |
|---|
| toString |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

# Constructor Detail

### ForcedDisconnectedEvent

`public` **`ForcedDisconnectedEvent`**`(VBIFilterGroup f)`

A constructor of this class.

**Parameters:**

`f` - Instance of a VBIFilterGroup that is the source of this event.

# Method Detail

### getSource

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

This method returns an instance of a class implementing VBIFilterGroup that is the source of the event.

**Overrides:**

`getSource` in class `org.davic.resources.ResourceStatusEvent`

**Returns:**

instance of a class implementing VBIFilterGroup that is the source of the event

**org.ocap.media**

## Interface MediaAccessAuthorization

```
public interface MediaAccessAuthorization
```

A `MediaAccessAuthorization` object represents the presentation authorization given by a registered MediaAccessHandler for a specific A/V content. When the `MediaAccessHandler` is triggered by the OCAP implementation, it returns a `MediaAccessAuthorization` to indicate which service components should not be presented, if any, with a list of reasons for denied access (use constant defined in `AlternativeMediaPresentationReason`) per service component.

**See Also:**

`MediaAccessHandler`, `AlternativeMediaPresentationReason`

# Method Summary

| | |
|---|---|
| int | **getDenialReasons**(org.davic.mpeg.ElementaryStream es)<br>Return a bit mask of denial reasons for the given service component. |
| java.util.Enumeration | **getDeniedElementaryStreams**()<br>Returns the list of service components whose presentation has not been authorized by the `MediaAccessHandler`. |
| boolean | **isFullAuthorization**()<br>Returns true if the presentation of all service components is authorized. |

# Method Detail

### isFullAuthorization

```
boolean isFullAuthorization()
```

Returns true if the presentation of all service components is authorized. False otherwise.

**Returns:**

true if the presentation of all service components is authorized. False otherwise.

### getDeniedElementaryStreams

```
java.util.Enumeration getDeniedElementaryStreams()
```

Returns the list of service components whose presentation has not been authorized by the `MediaAccessHandler`.

**Returns:**

an Enumeration of `ElementaryStream` whose presentation has not been authorized by the `MediaAccessHandler`.

### getDenialReasons

```
int getDenialReasons(org.davic.mpeg.ElementaryStream es)
```

Return a bit mask of denial reasons for the given service component. Denial reasons are defined in `AlternativeMediaPresentationReason`

**Parameters:**

`es` - the service component the `MediaAccessHandler` refused presentation.

**Returns:**

a bit mask of reasons for the denial. This bit mask is made of `AlternativeMediaPresentationReason`.

**Throws:**

`java.lang.IllegalArgumentException` - if the argument is not one of the objects in the enumeration returned by `getDeniedElementaryStreams()`.

**org.ocap.media**

## Interface MediaAccessConditionControl

**All Superinterfaces:**
javax.media.Control

```
public interface MediaAccessConditionControl
extends javax.media.Control
```

This interface allows an application to notify that conditions of media presentation in a running JMF player have been modified, and so the check for media presentation must be done. Instance of the MediaAccessConditionControl interface shall be obtained via a `Controller.getControl(java.lang.String)` and a `Controller.getControls()` method by all applications. But MonitorAppPermission("mediaAccess") is necessary to call methods in this interface.

# Method Summary

| | |
|---|---|
| void | **conditionHasChanged**(MediaPresentationEvaluationTrigger trigger)<br>Notifies the player that the conditions to authorize the service presentation have been modified, and so a new check must be done for the specified player. |

| Methods inherited from interface javax.media.Control |
|---|
| getControlComponent |

# Method Detail

### conditionHasChanged

```
void conditionHasChanged(MediaPresentationEvaluationTrigger trigger)
```

Notifies the player that the conditions to authorize the service presentation have been modified, and so a new check must be done for the specified player.

Registered {link MediaAccessHandler} will be called.

**Parameters:**
`trigger` - any of the optional trigger defined in `MediaPresentationEvaluationTrigger` or an application defined MediaPresentationEvaluationTrigger object.
**Throws:**
`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("mediaAccess")
**See Also:**
`MediaPresentationEvaluationTrigger`, `MediaAccessHandler`

**org.ocap.media**

## Interface MediaAccessHandler

```
public interface MediaAccessHandler
```

A class implementing this interface can prevent the presentation of A/V service components.

Only one instance of the class that implements this interface can be registered to `MediaAccessHandlerRegistrar` via the `MediaAccessHandlerRegistrar.registerMediaAccessHandler(MediaAccessHandler)` method. JMF calls checkMediaAccessAuthorization() before AV service components presentation.

An application which has a MonitorAppPermission("mediaAccess") may implement this interface, and may set an instance of it in `MediaAccessHandlerRegistrar`.

Note : this handler is only responsible for the presentation of A/V service components and not for the launching of applications.

# Method Summary

| | |
|---|---|
| `MediaAccessAuthorization` | **`checkMediaAccessAuthorization`**`(javax.media.Player p, OcapLocator sourceURL, boolean isSourceDigital, org.davic.mpeg.ElementaryStream[] esList, MediaPresentationEvaluationTrigger evaluationTrigger)`<br>The `checkMediaAccessAuthorization()` method is invoked each time a `MediaPresentationEvaluationTrigger` is generated either by the OCAP implementation, or, by a monitor application that has MonitorAppPermission("mediaAccess") through the `MediaAccessConditionControl` JMF control. |

# Method Detail

### checkMediaAccessAuthorization

```
MediaAccessAuthorization checkMediaAccessAuthorization(javax.media.Player p,
                                                       OcapLocator sourceURL,

boolean isSourceDigital,

org.davic.mpeg.ElementaryStream[] esList,

MediaPresentationEvaluationTrigger evaluationTrigger)
```

The `checkMediaAccessAuthorization()` method is invoked each time a `MediaPresentationEvaluationTrigger` is generated either by the OCAP implementation, or, by a monitor application that has MonitorAppPermission("mediaAccess") through the `MediaAccessConditionControl` JMF control. The OCAP implementation SHALL block the new presentation corresponding to the new environment that led to the generation of the trigger until the MediaAccessHandler grants permission. It is implementation dependent whether presentation of previously

selected service components is stopped or not. The OCAP implementation gives all the service components that are part of the service selection even if they are already presented before the trigger is issued.

**Parameters:**

`p` - the concerned player.

`sourceURL` - the URL of the content to be presented.

`isSourceDigital` - a boolean indicating if the source is digital or analog.

`esList` - is the list of service components that are going to be presented. esList can be null, for instance if isSourceDigital is false.

`evaluationTrigger` - is one of the constant defined in `MediaPresentationEvaluationTrigger` or an application defined `MediaPresentationEvaluationTrigger`.

**Returns:**

a `MediaAccessAuthorization` defined by MediaAccessHandler for the given service components. The MediaAccessAuthorization contains the reason(s), if any, of denied access (use constant defined in `AlternativeMediaPresentationReason`) per service component.

**See Also:**

`MediaAccessAuthorization`, `AlternativeMediaPresentationReason`, `MediaPresentationEvaluationTrigger`

**org.ocap.media**

## Class MediaAccessHandlerRegistrar

```
java.lang.Object
  └ org.ocap.media.MediaAccessHandlerRegistrar
```

```
public abstract class MediaAccessHandlerRegistrar
extends java.lang.Object
```

This class allows to register a handler that can prevent the presentation of A/V service components. This handler must be plugged in the service selection before the service components presentation.

Moreover, it allows the Monitor Application to handle some triggers. A trigger is an event that can lead to an evaluation of the presented media.

## Constructor Summary

| | |
|---|---|
| protected | **MediaAccessHandlerRegistrar**()<br>Constructor of MediaAccessHandlerRegistrar. |

## Method Summary

| | |
|---|---|
| static MediaAccessHandlerRegistrar | **getInstance**()<br>This method returns the sole instance of the MediaAccessHandlerRegistrar class. |
| abstract void | **registerMediaAccessHandler**(MediaAccessHandler mah)<br>Registers the handler that can prevent the presentation of A/V service components. |
| abstract void | **setExternalTriggers** MediaPresentationEvaluationTrigger[] triggers)<br>Defines the list of triggers that the Monitor Application wants to generate. |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

## Constructor Detail

**MediaAccessHandlerRegistrar**

protected **MediaAccessHandlerRegistrar**()

Constructor of MediaAccessHandlerRegistrar. An application must use the `getInstance()` method to create an instance.

# Method Detail

### getInstance

```
public static MediaAccessHandlerRegistrar getInstance()
```

This method returns the sole instance of the `MediaAccessHandlerRegistrar` class. The `MediaAccessHandlerRegistrar` instance is a singleton.

**Returns:**

The `MediaAccessHandlerRegistrar` instance.

### registerMediaAccessHandler

```
public abstract void registerMediaAccessHandler(MediaAccessHandler mah)
```

Registers the handler that can prevent the presentation of A/V service components.

At most, only one instance of MediaAccessHandler can be set. Multiple calls of this method replace the previous instance by a new one.

By default, no MediaAccessHandler is set, i.e. the MediaAccessHandler.checkMediaAccessAuthorization() method is not called.

**Parameters:**

`mah` - The MediaAccessHandler to set. If null is set, the MediaAccessHandler instance will be removed.

**Throws:**

`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("mediaAccess") permission.

**See Also:**

`MediaAccessHandler`

### setExternalTriggers

```
public abstract void
setExternalTriggers(MediaPresentationEvaluationTrigger[] triggers)
```

Defines the list of triggers that the Monitor Application wants to generate. Such triggers must be MediaPresentationEvaluationTriggers tagged as "OPTIONAL". When set, the OCAP implementation stops generating corresponding `MediaPresentationEvaluationTrigger`.

**Parameters:**

`triggers` - Array of MediaPresentationEvaluationTriggers generated by the Monitor Application.

**Throws:**

`java.lang.SecurityException` - if the caller does not have MonitorAppPermission("mediaAccess") permission.

`java.lang.IllegalArgumentException` - if one of the input triggers is "MANDATORY".

**See Also:**

`MediaPresentationEvaluationTrigger`

**org.ocap.media**
## Class MediaPresentationEvaluationTrigger

```
java.lang.Object
  └ org.ocap.media.MediaPresentationEvaluationTrigger
```

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

This class represents possible reasons to trigger an evaluation that leads to the generation of an `AlternativeMediaPresentationEvent` or a `NormalMediaPresentationEvent`. An application which has a MonitorAppPermission("mediaAccess") can use predefined `MediaPresentationEvaluationTrigger` or define its own `MediaPresentationEvaluationTrigger` and indicate to the implementation that presentation conditions have changed through the `MediaAccessConditionControl`.

MANDATORY triggers: OCAP implementation SHALL be able to generate such trigger independently of the monitor application. A monitor application cannot generate such triggers exclusively.
OPTIONAL triggers: such triggers MAY be generated by the OCAP implementation. A monitor application can exclusively generate such triggers.

**See Also:**
`MediaAccessHandler`

| Field Summary | |
|---|---|
| `static MediaPresentationEvaluationTrigger` | **`CURRENT_PROGRAM_EVENT_CHANGED`** `MediaPresentationEvaluationTrigger` indicating that current program event has changed. |
| `static MediaPresentationEvaluationTrigger` | **`NEW_SELECTED_SERVICE`** `MediaPresentationEvaluationTrigger` indicating that a new service has been selected. |
| `static MediaPresentationEvaluationTrigger` | **`NEW_SELECTED_SERVICE_COMPONENTS`** `MediaPresentationEvaluationTrigger` indicating that new service components have been selected via JMF control or via ServiceContext. |
| `static MediaPresentationEvaluationTrigger` | **`PMT_CHANGED`** `MediaPresentationEvaluationTrigger` indicating that the broadcast PMT has changed. |
| `static MediaPresentationEvaluationTrigger` | **`POWER_STATE_CHANGED`** `MediaPresentationEvaluationTrigger` indicating that the power state has changed, e.g., switch to Software Standby. |
| `static MediaPresentationEvaluationTrigger` | **`PROGRAM_EVENT_RATING_CHANGED`** `MediaPresentationEvaluationTrigger` indicating that program event rating has changed. |

# Field Summary

| | |
|---|---|
| `static MediaPresentationEvaluationTrigger` | **`RESOURCE_AVAILABILITY_CHANGED`** `MediaPresentationEvaluationTrigger` indicating that access to a resource has changed : lost or free resource. |
| `static MediaPresentationEvaluationTrigger` | **`USER_RATING_CHANGED`** `MediaPresentationEvaluationTrigger` indicating that the user preference for rating has been changed. |

# Constructor Summary

| | |
|---|---|
| `protected` | **`MediaPresentationEvaluationTrigger()`** Constructs a MediaPresentationEvaluationTrigger. |

# Method Summary

| | |
|---|---|
| `boolean` | **`isOptional()`** Returns true if the trigger can be generated either by the OCAP implementation or by the Monitor Application. |

| **Methods inherited from class java.lang.Object** |
|---|
| `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait` |

# Field Detail

**PMT_CHANGED**

`public static final MediaPresentationEvaluationTrigger` **`PMT_CHANGED`**
`MediaPresentationEvaluationTrigger` indicating that the broadcast PMT has changed.
MANDATORY trigger

**RESOURCE_AVAILABILITY_CHANGED**

`public static final MediaPresentationEvaluationTrigger`
**`RESOURCE_AVAILABILITY_CHANGED`**
`MediaPresentationEvaluationTrigger` indicating that access to a resource has changed : lost or free resource.
MANDATORY trigger

**NEW_SELECTED_SERVICE**

`public static final MediaPresentationEvaluationTrigger` **`NEW_SELECTED_SERVICE`**
`MediaPresentationEvaluationTrigger` indicating that a new service has been selected.
MANDATORY trigger

#### NEW_SELECTED_SERVICE_COMPONENTS

`public static final MediaPresentationEvaluationTrigger`
**`NEW_SELECTED_SERVICE_COMPONENTS`**
`MediaPresentationEvaluationTrigger` indicating that new service components have been selected via JMF control or via ServiceContext.
MANDATORY trigger

#### POWER_STATE_CHANGED

`public static final MediaPresentationEvaluationTrigger` **`POWER_STATE_CHANGED`**
`MediaPresentationEvaluationTrigger` indicating that the power state has changed, e.g., switch to Software Standby.
OPTIONAL trigger

#### CURRENT_PROGRAM_EVENT_CHANGED

`public static final MediaPresentationEvaluationTrigger`
**`CURRENT_PROGRAM_EVENT_CHANGED`**
`MediaPresentationEvaluationTrigger` indicating that current program event has changed.
OPTIONAL trigger

#### USER_RATING_CHANGED

`public static final MediaPresentationEvaluationTrigger` **`USER_RATING_CHANGED`**
`MediaPresentationEvaluationTrigger` indicating that the user preference for rating has been changed.
OPTIONAL trigger

#### PROGRAM_EVENT_RATING_CHANGED

`public static final MediaPresentationEvaluationTrigger`
**`PROGRAM_EVENT_RATING_CHANGED`**
`MediaPresentationEvaluationTrigger` indicating that program event rating has changed.
OPTIONAL trigger

## Constructor Detail

#### MediaPresentationEvaluationTrigger

`protected` **`MediaPresentationEvaluationTrigger`**`()`
Constructs a MediaPresentationEvaluationTrigger.

# Method Detail

### isOptional

`public boolean` **`isOptional`**`()`

Returns true if the trigger can be generated either by the OCAP implementation or by the Monitor Application. Returns false if the trigger is generated by the OCAP implementation.

**Returns:**

true if the trigger can be generated either by the implementation or by the Monitor Application. false if the trigger is generated by the OCAP implementation.

**org.ocap.media**

## Class MediaPresentationEvent

```
java.lang.Object
  └ java.util.EventObject
      └ javax.media.ControllerEvent
          └ javax.media.TransitionEvent
              └ org.ocap.media.MediaPresentationEvent
```

**All Implemented Interfaces:**

java.io.Serializable, javax.media.MediaEvent

**Direct Known Subclasses:**

AlternativeMediaPresentationEvent, NormalMediaPresentationEvent

```
public abstract class MediaPresentationEvent
extends javax.media.TransitionEvent
```

`MediaPresentationEvent` is a JMF event used as the parent class of events indicating dynamic changes to the presentation of media components.

This event provides the trigger that leads to the generation of this event.

# Field Summary

| Fields inherited from class java.util.EventObject |
|---|
| source |

# Constructor Summary

| | |
|---|---|
| protected | **MediaPresentationEvent**(javax.media.Controller from, int previous, int current, int target)<br>Constructor of MediaPresentationEvent |

# Method Summary

| | |
|---|---|
| org.davic.mpeg.ElementaryStream[] | **getPresentedStreams**() |
| OcapLocator | **getSourceURL**() |
| MediaPresentationEvaluationTrigger | **getTrigger**() |
| boolean | **isSourceDigital**() |

| Methods inherited from class javax.media.TransitionEvent |
| --- |
| getCurrentState, getPreviousState, getTargetState, toString |

| Methods inherited from class javax.media.ControllerEvent |
| --- |
| getSource, getSourceController |

| Methods inherited from class java.lang.Object |
| --- |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

# Constructor Detail

### MediaPresentationEvent

```
protected MediaPresentationEvent(javax.media.Controller from,
                                 int previous,
                                 int current,
                                 int target)
```

Constructor of MediaPresentationEvent

**See Also:**
`TransitionEvent`

# Method Detail

### isSourceDigital

```
public boolean isSourceDigital()
```

**Returns:**
Returns true if the presented source is digital, false otherwise.

### getSourceURL

```
public OcapLocator getSourceURL()
```

**Returns:**
Returns the URL of the source that is presented.

### getPresentedStreams

```
public org.davic.mpeg.ElementaryStream[] getPresentedStreams()
```

**Returns:**
Returns the service components that are currently presented. If no service components is presented or the source is analog, null is returned.

### getTrigger

```
public MediaPresentationEvaluationTrigger getTrigger()
```

**Returns:**
Returns the trigger that leads to the generation of a MediaPresentationEvent.

**See Also:**
`MediaPresentationEvaluationTrigger`

**org.ocap.media**
## Class MediaTimer

```
java.lang.Object
  └ org.ocap.media.MediaTimer
```

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

This is a timer class that counts time based on a media time of a specified Player. A media time is specified by the JMF specification. An application can specify a range between a first time and a last time. When a current media time exceeds this range, this timer fires and calls a MediaTimerListener.notify() method. I.e., when a current media time passes the last time, this timer fires and calls the notify() method with MEDIA_WENTOFF_LAST event. On the other hand, when a current media time passes the first time in a reverse playback or a skip playback, this timer fires and calls the notify() method with MEDIA_WENTOFF_FIRST event.

## Constructor Summary

| |
|---|
| **MediaTimer**(javax.media.Player p, MediaTimerListener listener)<br>Constructor to make a MediaTimer object that counts time based on the media time line of the specified Player. |

## Method Summary

| | |
|---|---|
| javax.media.Time | **getFirstTime**()<br>Get a first time that was set to this MediaTimer object. |
| javax.media.Time | **getLastTime**()<br>Get a last time that was set to this MediaTimer object. |
| javax.media.Player | **getPlayer**()<br>Get a Player that was tied to this MediaTimer object by a constructor. |
| void | **setFirstTime**(javax.media.Time time)<br>Set a first time of a time range. |
| void | **setLastTime**(javax.media.Time time)<br>Set a last time of a time range. |
| void | **start**()<br>Start this MediaTimer object. |
| void | **stop**()<br>Stop this MediaTimer object. |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Constructor Detail

### MediaTimer

```
public MediaTimer(javax.media.Player p,
                  MediaTimerListener listener)
```

Constructor to make a MediaTimer object that counts time based on the media time line of the specified Player.

**Parameters:**

`p` - a JMF Player.

# Method Detail

### setFirstTime

```
public void setFirstTime(javax.media.Time time)
```

Set a first time of a time range. This MediaTimer object shall go off when a current time passes the specified first time in a reverse playback or a skip playback. A first time value specified in the past will be cleared, when a new first time is set by this method.

**Parameters:**

`time` - a time to go off.

### setLastTime

```
public void setLastTime(javax.media.Time time)
```

Set a last time of a time range. This MediaTimer object shall go off when a current time passes the specified last time in a normal playback or a skip playback. A last time value specified in the past will be cleared, when a new last time is set by this method.

**Parameters:**

`time` - a time to go off.

### getFirstTime

```
public javax.media.Time getFirstTime()
```

Get a first time that was set to this MediaTimer object.

**Returns:**

a time to go off.

### getLastTime

```
public javax.media.Time getLastTime()
```

Get a last time that was set to this MediaTimer object.

**Returns:**

a time to go off.

### start

```
public void start()
```

Start this MediaTimer object. A MediaTimerListener is called with TIMER_START event value by the OCAP implementation.

#### stop

```
public void stop()
```

Stop this MediaTimer object. A MediaTimerListener is called with TIMER_STOP event value by the OCAP implementation.

#### getPlayer

```
public javax.media.Player getPlayer()
```

Get a Player that was tied to this MediaTimer object by a constructor.

**Returns:**

a Player that was tied to this MediaTimer object.

**org.ocap.media**
## Interface MediaTimerListener

```
public interface MediaTimerListener
```

This is a listener to inform events on a MediaTimer object.

# Field Summary

| | |
|---|---|
| static int | **TIMER_START**<br>Indicates a MediaTimer starts. |
| static int | **TIMER_STOP**<br>Indicates a MediaTimer stops. |
| static int | **TIMER_WENTOFF_FIRST**<br>Indicates a MediaTimer went off since a current media time passes the specified first time in a reverse playback or a skip playback. |
| static int | **TIMER_WENTOFF_LAST**<br>Indicates a MediaTimer went off since a current media time passes the specified last time in a normal playback or a skip playback. |

# Method Summary

| | |
|---|---|
| void | **notify**(int event, javax.media.Player p)<br>This is a call back method to inform event on a MediaTimer. |

# Field Detail

### TIMER_START

```
static final int TIMER_START
```
Indicates a MediaTimer starts.

### TIMER_STOP

```
static final int TIMER_STOP
```
Indicates a MediaTimer stops.

### TIMER_WENTOFF_FIRST

```
static final int TIMER_WENTOFF_FIRST
```
Indicates a MediaTimer went off since a current media time passes the specified first time in a reverse playback or a skip playback.

### TIMER_WENTOFF_LAST

```
static final int TIMER_WENTOFF_LAST
```

Indicates a MediaTimer went off since a current media time passes the specified last time in a normal playback or a skip playback.

# Method Detail

### notify

```
void notify(int event,
            javax.media.Player p)
```

This is a call back method to inform event on a MediaTimer.

**Parameters:**

`event` - an event value that happened on a MediaTimer. One of constants that have TIMER_ prefix.

`p` - a Player that was tied to this MediaTimer object.

**org.ocap.media**

## Class NormalMediaPresentationEvent

```
java.lang.Object
  └ java.util.EventObject
      └ javax.media.ControllerEvent
          └ javax.media.TransitionEvent
              └ org.ocap.media.MediaPresentationEvent
                  └ org.ocap.media.NormalMediaPresentationEvent
```

**All Implemented Interfaces:**

java.io.Serializable, javax.media.MediaEvent

```
public abstract class NormalMediaPresentationEvent
extends MediaPresentationEvent
```

`NormalMediaPresentationEvent` is a JMF event generated when the normal media components of a service are presented.

Media presentation is considered as normal when explicitly selected service components (by a dedicated API), or implicitly selected service components (by the player itself) can be presented to the user. Media presentation is considered as "alternative" in any other case, especially when it is caused by one of the reasons described in `AlternativeMediaPresentationReason`.

`NormalMediaPresentationEvent` notification is generated :

- When normal media content presentation begins;
- During the presentation of a service, if alternative media content was presented and all of that media alternative content is replaced by a content which is a normal part of the service concerned;
- During the presentation of a service, if normal media content was being presented and an evaluation leads to a new normal media content presentation.

# Field Summary

| Fields inherited from class java.util.EventObject |
|---|
| source |

# Constructor Summary

| | |
|---|---|
| protected | **NormalMediaPresentationEvent**(javax.media.Controller from, int previous, int current, int target)<br>Constructor of MediaPresentationEvent |

# Method Summary

| Methods inherited from class org.ocap.media.MediaPresentationEvent |
| --- |
| getPresentedStreams, getSourceURL, getTrigger, isSourceDigital |

| Methods inherited from class javax.media.TransitionEvent |
| --- |
| getCurrentState, getPreviousState, getTargetState, toString |

| Methods inherited from class javax.media.ControllerEvent |
| --- |
| getSource, getSourceController |

| Methods inherited from class java.lang.Object |
| --- |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

# Constructor Detail

### NormalMediaPresentationEvent

```
protected NormalMediaPresentationEvent(javax.media.Controller from,
                                       int previous,
                                       int current,
                                       int target)
```

Constructor of MediaPresentationEvent

**See Also:**
MediaPresentationEvent

**org.ocap.media**

## Interface NotPresentedMediaInterface

**All Known Implementing Classes:**
AlternativeMediaPresentationEvent

```
public interface NotPresentedMediaInterface
```

NotPresentedMediaInterface shall be implemented by classes which can report failure to access media components. The interface provides an ability for an application to find out the list of not presented service components and some information about the reason for the failure.

# Method Summary

| | |
|---|---|
| org.davic.mpeg.ElementaryStream[] | **getNotPresentedStreams**() |
| int | **getReason**(org.davic.mpeg.ElementaryStream es) |

# Method Detail

### getNotPresentedStreams

```
org.davic.mpeg.ElementaryStream[] getNotPresentedStreams()
```

**Returns:**
Returns the subset of explicitly (by Application request) or implicitly (by the Player itself) service components that were selected and which presentation was not possible.

### getReason

```
int getReason(org.davic.mpeg.ElementaryStream es)
```

**Parameters:**
es - a not presented service component.
**Returns:**
Returns a bit mask of reasons that lead to the non presentation of the given service component. The reasons are defined in AlternativeMediaPresentationReason)interface.

**org.ocap.media**

## Interface VBIFilter

```
public interface VBIFilter
```

This class represents a VBI filter. A VBIFilter instances are created by a VBIFilterGroup based on the OCAP resource management. Line numbers and a data format to be filtered are specified when the filter is created.

The startFiltering() method starts filtering of the specified data format in the specified VBI line and stores data units in an internal buffer. When the first single data unit is filtered, a VBIFilterEvent with EVENT_CODE_FIRST_VBI_DATA_AVAILABLE is issued only once. The VBIFilter continues filtering.

VBI filtering stops in the following cases:

- If a stopFiltering() method is called, a VBIFilterEvent with EVENT_CODE_FORCIBLE_TERMINATED notifies it.
- If an internal buffer is full, a VBIFilterEvent with EVENT_CODE_BUFFER_FULL notifies it.
- If a time out (specified by a timeout value in a `setTimeOut(long)`) occurs, a VBIFilterEvent with EVENT_CODE_TIMEOUT notifies it.

VBI filtering continues in the following cases:

- Timer notification by a VBIFilterEvent with EVENT_CODE_TIME_NOTIFICATION.
- Cyclic notification for every specified number of data units by a VBIFilterEvent with EVENT_CODE_UNITS_NOTIFICATION.

**Author:**
Shigeaki Watanabe (Panasonic)

# Method Summary

| | |
|---|---|
| void | **addVBIFilterListener**(VBIFilterListener listener)<br>Add a new VBIFilterListener instance to this VBI filter. |
| void | **clearBuffer**()<br>Clear an internal buffer to store retrieved VBI data. |
| byte[] | **getVBIData**()<br>This method returns multiple VBI data unit bytes. |
| void | **removeVBIFilterListener**(VBIFilterListener listener)<br>Remove an existing VBIFilterListener instance from this VBI filter. |
| void | **setNotificationByDataUnits**(int numberOfDataUnits)<br>Set the number of data units to reveive a cyclic notification. |
| void | **setNotificationByTime**(long milliseconds)<br>Set a notification time. |
| void | **setTimeOut**(long milliseconds)<br>Set a timeout value. |

# Method Summary

| | |
|---|---|
| void | **startFiltering**(java.lang.Object appData)<br>Initiate filtering of VBI data for the specified line and the specified data format by a VBIFilterGroup. |
| void | **startFiltering**(java.lang.Object appData, int offset, byte[] posFilterDef, byte[] posFilterMask, byte[] negFilterDef, byte[] negFilterMask)<br>Initiate filtering of VBI data for the specified line and the specified data format by a VBIFilterGroup. |
| void | **stopFiltering**()<br>Stop current filtering of this VBI filter. |

# Method Detail

### startFiltering

```
void startFiltering(java.lang.Object appData)
```

Initiate filtering of VBI data for the specified line and the specified data format by a VBIFilterGroup. Filtering starts only after the `VBIFilterGroup.attach(javax.tv.service.selection.ServiceContext, org.davic.resources.ResourceClient, java.lang.Object)` method is called.

**Parameters:**

`appData` - application specific data. This data is notified to the application with a SectionFilterEvent. Null is possible.

---

### startFiltering

```
void startFiltering(java.lang.Object appData,
                    int offset,
                    byte[] posFilterDef,
                    byte[] posFilterMask,
                    byte[] negFilterDef,
                    byte[] negFilterMask)
```

Initiate filtering of VBI data for the specified line and the specified data format by a VBIFilterGroup. Only data unit(s) matching with a specified filter parameters are retrieved. Filtering starts only after the `VBIFilterGroup.attach(javax.tv.service.selection.ServiceContext, org.davic.resources.ResourceClient, java.lang.Object)` method is called.

**Parameters:**

`appData` - application specific data. This data is notified to the application with a SectionFilterEvent. Null is possible.

`offset` - defines a number of offset bytes that the specified matching bits and masking bits are applied. Value 0 means no offset. Value 1 means that the matching/masking bit is applied from the second byte.

`posFilterDef` - defines values to match for bits in a single data unit. Only data unit that has matching bytes with this posFilterDef are retrieved. Maximum length is 36 bytes.

`posFilterMask` - defines which bits in the data unit are to be compared against the posFilterDef bytes. Matching calculation of negFilterDef and negFilterMask obeys E.8.1 of DAVIC 1.4.1 Part 9. Maximum length is 36 bytes.

`negFilterDef` - defines values to match for bits in a single data unit. Only data unit that has matching bytes with this negFilterDef are retrieved. Maximum length is 36 bytes.

`negFilterMask` - defines which bits in the data unit are to be compared against the negFilterDef bytes. Matching calculation of negFilterDef and negFilterMask obeys E.8.1 of DAVIC 1.4.1 Part 9. Maximum length is 36 bytes.

#### stopFiltering

`void` **`stopFiltering`**`()`

Stop current filtering of this VBI filter. Note that the VBIFilterGroup holding this VBI filter doesn't detach.

#### setTimeOut

`void` **`setTimeOut`**`(long milliseconds)`

Set a timeout value. If no VBI data unit is retrieved after calling the startFiltering() method within the timeout value, the filtering stops automatically and SectionFilterEvent with EVENT_CODE_TIMEOUT notifies a timeout occured.

**Parameters:**

`milliseconds` - a timeout value in milli seconds. A default value is -1 that indicates infinite.

#### setNotificationByTime

`void` **`setNotificationByTime`**`(long milliseconds)`

Set a notification time. By setting a notification time, the OCAP implementation notifies a VBIFilterEvent with EVENT_CODE_TIME_NOTIFICATION when the specified time-period has elapsed after receiving the first byte of the data unit. The event shall be sent even if the data received does not form a complete data unit. The event is sent only once. The filter continues filtering after sending the event.

**Parameters:**

`milliseconds` - a time-period value in milli seconds. A default value is -1 that indicates infinite.

#### setNotificationByDataUnits

`void` **`setNotificationByDataUnits`**`(int numberOfDataUnits)`

Set the number of data units to reveive a cyclic notification. By setting the number of data units, the OCAP implementation notifies a VBIFilterEvent with EVENT_CODE_UNITS_NOTIFICATION cyclically everytime when the specified number of new data units are filtered and stored in a buffer. The filter continues filtering after sending the event.

**Parameters:**

`numberOfDataUnits` - the number of data units to be notified. A default value is 0 that indicates no notification. Note that if a small number of data units is specified, the notification may be delayed and affects to the host performance. For example, if 1 is specified for UNKNOWN data unit that comes every field (i.e., 1/60 seconds), the host has to notify every 1/60 seconds and makes an over load.

**Throws:**

`java.lang.IllegalArgumentException` - if the numberOfDataUnit is larger than the bufferSize specified by a `VBIFilterGroup.newVBIFilter(int[], int, int, int, int)` method.

#### addVBIFilterListener

`void` **`addVBIFilterListener`**`(VBIFilterListener listener)`

Add a new VBIFilterListener instance to this VBI filter. If the same instance that exists currently is specified, this method does nothing and no exception is thrown.

**Parameters:**

`listener` - a VBIFilterListener instance to be notified a VBI filtering events.

#### removeVBIFilterListener

`void` **`removeVBIFilterListener`**`(VBIFilterListener listener)`

Remove an existing VBIFilterListener instance from this VBI filter. If the specified instance has not been added, this method does nothing and no exception is thrown.

#### getVBIData

`byte[]` **`getVBIData`**`()`

This method returns multiple VBI data unit bytes. The data unit format is defined in a description of a `VBIFilter` interface. The returned bytes is a simple concatenated VBI data at the moment. Note that the return value is not aligned by a complete VBI data unit. I.e., incomplete data unit may return. When this method is called, an internal buffer is cleared once. I.e., next call returns a next byte of retrieved VBI data.

**Returns:**

a concatenated VBI data of the form of specified by a `VBIFilterGroup.newVBIFilter(int[], int, int, int, int)` method.

#### clearBuffer

`void` **`clearBuffer`**`()`

Clear an internal buffer to store retrieved VBI data. An application shall call this method before data full.

**org.ocap.media**

## Class VBIFilterEvent

```
java.lang.Object
  └ org.ocap.media.VBIFilterEvent
```

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

This class represents a VBI filter event. When a specific event happens, the `VBIFilterListener.filterUpdate(org.ocap.media.VBIFilterEvent)` method is called with an event that has a proper event code to indicate the event.

## Field Summary

| | |
|---|---|
| static int | **EVENT_CODE_BUFFER_FULL**<br>Indicates an internal buffer is full. |
| static int | **EVENT_CODE_FAILED_TO_DESCRAMBLE**<br>Indicates descrambling is unavailable for current video. |
| static int | **EVENT_CODE_FIRST_VBI_DATA_AVAILABLE**<br>Indicates that the first VBI data unit is available. |
| static int | **EVENT_CODE_FORCIBLE_TERMINATED**<br>Indicates current filtering is terminated forcibly for any reason except other EVENT_CODE_ constants. |
| static int | **EVENT_CODE_TIME_NOTIFICATION**<br>Indicates that a specified time-period elapsed after receiving the first byte of a data unit. |
| static int | **EVENT_CODE_TIMEOUT**<br>Indicates a timeout (specified by `VBIFilter.setTimeOut(long)`) occurred. |
| static int | **EVENT_CODE_UNITS_NOTIFICATION**<br>Indicates that the specified number of new data units are filtered and stored in a buffer cyclically. |
| static int | **EVENT_CODE_VIDEO_SOURCE_CHANGED**<br>Indicates the current video for VBI data unit filtering has changed. |

## Constructor Summary

| |
|---|
| **VBIFilterEvent()**<br>Constructor of this class. |

# Method Summary

| | |
|---|---|
| java.lang.Object | **getAppData**()<br>This method returns application specific data that was specified by VBIFilter.startFiltering() methods. |
| int | **getEventCode**()<br>This method returns the specific event code that caused this event. |
| java.lang.Object | **getSource**()<br>This method returns an instance of a class implementing VBIFilter that is the source of the event. |

| **Methods inherited from class java.lang.Object** |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Field Detail

### EVENT_CODE_FIRST_VBI_DATA_AVAILABLE

public static final int **EVENT_CODE_FIRST_VBI_DATA_AVAILABLE**

Indicates that the first VBI data unit is available. This event code is issued only once after calling `VBIFilter.startFiltering(java.lang.Object)` method even if multiple lines/fields is specified to the filter. Filtering continues.

### EVENT_CODE_FORCIBLE_TERMINATED

public static final int **EVENT_CODE_FORCIBLE_TERMINATED**

Indicates current filtering is terminated forcibly for any reason except other EVENT_CODE_ constants. E.g., a `VBIFilter.stopFiltering()` is called.

### EVENT_CODE_VIDEO_SOURCE_CHANGED

public static final int **EVENT_CODE_VIDEO_SOURCE_CHANGED**

Indicates the current video for VBI data unit filtering has changed. Note that current filtering doesn't stop even if this event happens. An application may stop filtering and then restart to retrieve valid data units.

### EVENT_CODE_FAILED_TO_DESCRAMBLE

public static final int **EVENT_CODE_FAILED_TO_DESCRAMBLE**

Indicates descrambling is unavailable for current video. Note that current filtering doesn't stop even if this event happens. Continues filtering until timeout.

### EVENT_CODE_TIMEOUT

public static final int **EVENT_CODE_TIMEOUT**

Indicates a timeout (specified by `VBIFilter.setTimeOut(long)`) occurred. I.e., this event code indicates no data unit is available.

#### EVENT_CODE_BUFFER_FULL

`public static final int` **`EVENT_CODE_BUFFER_FULL`**
Indicates an internal buffer is full. Filtering stops automatically.

#### EVENT_CODE_TIME_NOTIFICATION

`public static final int` **`EVENT_CODE_TIME_NOTIFICATION`**
Indicates that a specified time-period elapsed after receiving the first byte of a data unit.

#### EVENT_CODE_UNITS_NOTIFICATION

`public static final int` **`EVENT_CODE_UNITS_NOTIFICATION`**
Indicates that the specified number of new data units are filtered and stored in a buffer cyclically.

## Constructor Detail

#### VBIFilterEvent

`public` **`VBIFilterEvent`**`()`
Constructor of this class.

## Method Detail

#### getSource

`public java.lang.Object` **`getSource`**`()`
This method returns an instance of a class implementing VBIFilter that is the source of the event.
**Returns:**
instance of a class implementing VBIFilter that is the source of the event

#### getAppData

`public java.lang.Object` **`getAppData`**`()`
This method returns application specific data that was specified by VBIFilter.startFiltering() methods.
**Returns:**
an application specific data that was specified by VBIFilter.startFiltering() methods.

#### getEventCode

`public int` **`getEventCode`**`()`
This method returns the specific event code that caused this event.
**Returns:**
an event code. One of the constants that has EVENT_CODE_ prefix.

### org.ocap.media
## Class VBIFilterGroup

```
java.lang.Object
  └ org.ocap.media.VBIFilterGroup
```

**All Implemented Interfaces:**

org.davic.resources.ResourceProxy, org.davic.resources.ResourceServer

```
public class VBIFilterGroup
extends java.lang.Object
implements org.davic.resources.ResourceProxy,
org.davic.resources.ResourceServer
```

This class represents a group of VBI data filters. The VBI filters in a group are attached to a ServiceContext. Current video played back on the ServiceContext is the target of filtering. If the current video is analog, the filters retrieve specific data in a specified VBI line of the video. The data format is defined in each VBI data specification. If the current video is MPEG video, the filters retrieve specific data in the user_data found in the MPEG picture headers of the video. Data format in user_data is defined in ANSI/SCTE 20 and ANSI/SCTE 21.

When a new VBIFilterGroup is constructed, an application specifies the total number of VBIFilters held by it. The application can create a specified number of new VBIFilter instances via a `newVBIFilter(int[], int, int, int, int)` method. VBI lines and data format for filtering are specified in this method. After configuring all filters, the VBIFilterGroup is attached to a specific ServiceContext. Current video on the ServiceContext is the target of filtering. If the startFiltering() method of a VBI filter is called, the filter starts filtering immediately. The application can retrieve VBI data via a `VBIFilter.getVBIData()` method. The host may be able to filter VBI data using separated field (specify either filed 1 or field 2 and retrieve data in only the specified field) or by using mixed fields (retrieve data in both field 1 and 2 in arrival order).

The data unit is defined as follows.

**XDS (VBI_DATA_FORMAT_XDS)**
XDS packets in line 21 field 2 is defined in EIA-608-B. It has a start/continue/end code and is interleaved with CC3, CC4, T3 and T4 service. One XDS packet is a single data unit of this data format.

**Text service (VBI_DATA_FORMAT_T1/T2/T3/T4)**
Text service data in line 21 field 1 or 2 is defined in EIA-608-B. It has start/end codes and is interleaved with the other closed captioning service and Text service according to Section 7 of EIA-608-B. A single data unit of Text service is a character sequence between a text mode in code (RTD or TR) and a text mode out codes (EOC, RCL, RDC, RU2, RU3 or RU4). Note that Text service data unit returned from VBIFilter doesn't contain these text mode in/out code, but contain another control code like EDM etc.

**Generic EIA-608-B service (VBI_DATA_FORMAT_EIA608B)**
Generic EIA-608-B service represents a character stream that consists of Character One and Character Two in a field of line 21. Character One and Character Two are defined in Figure 1 of EIA-608-B. Note that the VBIFilter only supports separated field filtering since Character One and Character Two makes a separated data stream in each field. A single data unit of this format is a set of Character One and Character Two in one VBI line and field (i.e., only two characters).

**NABTS (VBI_DATA_FORMAT_NABTS)**
NABTS is defined in EIA-516. It is a 33 byte fixed length packet and consists of 5 byte prefix and 28 byte data block. Note that the filter retrieves only the "Data packet" part. I.e., one NABTS packet is a single data unit. The VBIFilter retrieves multiple NABTS packets in arrival order for all specified lines and fields.

**AMOL I/II (VBI_DATA_FORMAT_AMOL_I/II)**
AMOL I and II is defined in ACN 403-1218-024. It specifies header bits and the number of bits in a VBI line . The bits in one VBI line/field is a single data unit for this data format, i.e., the header bits are included in the data unit.

**UNKNOWN (VBI_DATA_FORMAT_UNKNOWN)**
UNKNOWN represents a VBI waveform that is unknown. To retrieve UNKNOWN data unit, the OCAP implementation needs to synchronize to the VBI's bit rate automatically. It is not guaranteed to success filtering even if the OCAP implementation supports UNKNOWN format, since an analog waveform is not standardized. The bits in one VBI line and field is a data unit for UNKNOWN format.

OCAP 1.0 hosts shall support retrieval of CC1, CC2, T1 and T2 data in VBI line 21 field 1 and CC3, CC4, T3, T4 and XDS in VBI line 21 field 2. Filtering of these caption and text mode service data is not provided to display captioning text on a screen synchronizing with video and audio. It is not guaranteed that filtering performance satisfies expected delay.

Support of the other VBI lines are optional. And support of the ANSI/SCTE 20 and ANSI/SCTE 21 is also optional. An application can check the supported filtering lines, data format and filtering techniques (field separated or field mixed) via capability methods.

# Field Summary

| | |
|---|---|
| `static int` | **`FIELD_1`**<br>Represents filtering of VBI data in field 1 only. |
| `static int` | **`FIELD_2`**<br>Represents filtering of VBI data in field 2 only. |
| `static int` | **`FIELD_MIXED`**<br>Represents filtering of VBI data in both field 1 and field 2 in arrival order. |
| `static int` | **`VBI_DATA_FORMAT_AMOL_I`**<br>Represents an AMOL I data format defined by ACN 403-1218-024. |
| `static int` | **`VBI_DATA_FORMAT_AMOL_II`**<br>Represents an AMOL II data format defined by ACN 403-1218-024. |
| `static int` | **`VBI_DATA_FORMAT_EIA608B`**<br>Represents a EIA-608-B generic Character One and Character Two format of line 21 field 1 or 2. |
| `static int` | **`VBI_DATA_FORMAT_NABTS`**<br>Represents an NABTS data format defined by EIA-516. |
| `static int` | **`VBI_DATA_FORMAT_T1`**<br>Represents a Text service T1 format of line 21 field 1. |
| `static int` | **`VBI_DATA_FORMAT_T2`**<br>Represents a Text service T2 format of line 21 field 1. |
| `static int` | **`VBI_DATA_FORMAT_T3`**<br>Represents a Text service T3 format of line 21 field 2. |
| `static int` | **`VBI_DATA_FORMAT_T4`**<br>Represents a Text service T4 format of line 21 field 2. |
| `static int` | **`VBI_DATA_FORMAT_UNKNOWN`**<br>Represents an unknown. |
| `static int` | **`VBI_DATA_FORMAT_XDS`**<br>Represents a XDS data format of line 21 field 2. |

## Constructor Summary

| |
|---|
| **VBIFilterGroup**(int numberOfFilters)<br>Creates a VBIFilterGroup instance that includes the specified number of VBI filters. |

## Method Summary

| | |
|---|---|
| void | **addResourceStatusEventListener**(org.davic.resources.ResourceStatusListener listener)<br>Add a listener object to be notified of changes in the status of resources related to VBI filters. |
| void | **attach**(javax.tv.service.selection.ServiceContext serviceContext, org.davic.resources.ResourceClient client, java.lang.Object requestData)<br>This method attempts to reserve all VBI filters held by this VBIFilterGroup, and attaches to a specified ServiceContext. |
| void | **detach**()<br>Release all filters in this VBIFilterGroup and terminate the connection to the ServiceContext. |
| org.davic.resources.ResourceClient | **getClient**()<br>Returns the ResourceClient object specified in the attach() method. |
| boolean | **getMixedFilteringCapability**(int[] lineNumber, int dataFormat)<br>Returns true if field mixed filtering of the specified VBI line numbers with the specified data format is supported by this VBIFilterGroup. |
| boolean | **getSCTE20Capability**()<br>Indicates if the host supports line 21 VBI data retrieval from user_data in MPEG picture headers as defined in ANSI/SCTE 20. |
| boolean | **getSCTE21Capability**()<br>Indicates if the host supports line 21 VBI data retrieval from user_data in MPEG picture headers as defined in ANSI/SCTE 21. |
| boolean | **getSeparatedFilteringCapability**(int[] lineNumber, int dataFormat)<br>Returns true if field separated filtering of the specified VBI line numbers with the specified data format is supported by this VBIFilterGroup. |
| javax.tv.service.selection.ServiceContext | **getServiceContext**()<br>Returns a ServiceContext instance specified by a attach(javax.tv.service.selection.ServiceContext, org.davic.resources.ResourceClient, java.lang.Object) method. |
| VBIFilter | **newVBIFilter**(int[] lineNumber, int field, int dataFormat, int unitLength, int bufferSize)<br>Create a new VBIFilter instance within a VBIFilterGroup instance. |
| void | **removeResourceStatusEventListener**(org.davic.resources.ResourceStatusListener listener)<br>Remove a specified listener object from this VBIFilterGroup. |

| Methods inherited from class java.lang.Object |
| --- |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Field Detail

### VBI_DATA_FORMAT_XDS

```
public static final int VBI_DATA_FORMAT_XDS
```
Represents a XDS data format of line 21 field 2.

### VBI_DATA_FORMAT_T1

```
public static final int VBI_DATA_FORMAT_T1
```
Represents a Text service T1 format of line 21 field 1.

### VBI_DATA_FORMAT_T2

```
public static final int VBI_DATA_FORMAT_T2
```
Represents a Text service T2 format of line 21 field 1.

### VBI_DATA_FORMAT_T3

```
public static final int VBI_DATA_FORMAT_T3
```
Represents a Text service T3 format of line 21 field 2.

### VBI_DATA_FORMAT_T4

```
public static final int VBI_DATA_FORMAT_T4
```
Represents a Text service T4 format of line 21 field 2.

### VBI_DATA_FORMAT_EIA608B

```
public static final int VBI_DATA_FORMAT_EIA608B
```
Represents a EIA-608-B generic Character One and Character Two format of line 21 field 1 or 2.

### VBI_DATA_FORMAT_NABTS

```
public static final int VBI_DATA_FORMAT_NABTS
```
Represents an NABTS data format defined by EIA-516.

### VBI_DATA_FORMAT_AMOL_I

```
public static final int VBI_DATA_FORMAT_AMOL_I
```

Represents an AMOL I data format defined by ACN 403-1218-024.

#### VBI_DATA_FORMAT_AMOL_II

```
public static final int VBI_DATA_FORMAT_AMOL_II
```
Represents an AMOL II data format defined by ACN 403-1218-024.

#### VBI_DATA_FORMAT_UNKNOWN

```
public static final int VBI_DATA_FORMAT_UNKNOWN
```
Represents an unknown. I.e., all data bits in the VBI line are retrieved.

#### FIELD_1

```
public static final int FIELD_1
```
Represents filtering of VBI data in field 1 only.

#### FIELD_2

```
public static final int FIELD_2
```
Represents filtering of VBI data in field 2 only.

#### FIELD_MIXED

```
public static final int FIELD_MIXED
```
Represents filtering of VBI data in both field 1 and field 2 in arrival order.

# Constructor Detail

#### VBIFilterGroup

```
public VBIFilterGroup(int numberOfFilters)
```
Creates a VBIFilterGroup instance that includes the specified number of VBI filters. The VBI filters are reserved when the `attach(javax.tv.service.selection.ServiceContext, org.davic.resources.ResourceClient, java.lang.Object)` method is called.

**Parameters:**

`numberOfFilters` - the number of requested VBI filters held by a new VBIFilterGroup instance.

**Throws:**

`java.lang.SecurityException` - if the caller doesn't have MonitorAppPermission("vbifiltering").

# Method Detail

#### newVBIFilter

```
public VBIFilter newVBIFilter(int[] lineNumber,
                              int field,
                              int dataFormat,
                              int unitLength,
```

```
                                    int bufferSize)
```

Create a new VBIFilter instance within a VBIFilterGroup instance. The VBIFilter filters the specified VBI line/field and retrieves data units of the specified format. The filter has an internal buffer of the specified size to store the filtered data units. If the filtered data size exceeds the buffer size, the oversized part of data will be lost.

**Parameters:**

`lineNumber` - an array of VBI line numbers that will be filtered by a returned SimpleVBIFilter. If line 21 is specified, the filter also filters user_data defined in ANSI/SCTE 20 and ANSI/SCTE 21, when a current video is MPEG. Note that filtering of user_data is optional.

`field` - a field number that will be filtered by a returned VBIFilter. One of the constants that has a FIELD_ prefix.

If FIELD_MIXED is specified, the VBIFilter filters both field 1 and 2 of the specified lineNumber lines. Filtered data is concatenated in arrival order.

If FIELD_1 is specified, the VBIFilter filters only field 1 of the specified lineNumber lines.

If FIELD_2 is specified, the VBIFilter filters only field 2 of the specified lineNumber lines.

`dataFormat` - one of constants that has a VBI_DATA_FORMAT_ prefix to specify a data format to be filtered.

`unitLength` - specify the number of bits (not bytes) of a single data unit for the VBI_DATA_FORMAT_UNKNOWN format. This value shall be ignored for the other well known format.

`bufferSize` - number of bytes to specify size of an internal buffer used to hold a VBI data units.

**Returns:**

a new VBIFilter instance. Total number of filters created by a VBIFilterGroup shall not exceed the number of filters specified by a constructor.

**Throws:**

`java.lang.IllegalStateException` - if the total number of filters created by this VBIFilterGroup exceeds the specified number by a constructor.

`java.lang.IllegalArgumentException` - if this VBIFilterGroup can't create a VBIFilter for specified parameters due to hardware or software restrictions, or the lineNumber is not capable line, or if the field is not defined in constants, or the specified dataFormat is not defined in constants, or if the bufferSize is 0. For example, if this method is called with lineNumber=21, field=FIELD_1 and dataFormat=VBI_DATA_FORMAT_XDS, this exception is thrown and null returns.

### attach

```
public void attach(javax.tv.service.selection.ServiceContext serviceContext,
                   org.davic.resources.ResourceClient client,
                   java.lang.Object requestData)
```

This method attempts to reserve all VBI filters held by this VBIFilterGroup, and attaches to a specified ServiceContext. After the method call, the filter starts filtering of VBI data units contained in current selected video on the ServiceContext responding to a `VBIFilter.startFiltering(java.lang.Object)` call. If the startFiltering method has already been called, the filter starts filtering immediately. Note that the number of filters specified by a constructor shall be created by the newVBIFilter() method before calling this method. (I.e., VBI lines and a data format shall be specified before resource reservation.)

**Parameters:**

`serviceContext` - a ServiceContext that selects the video to be filtered. It is not necessary that a specified ServiceContext is in the presenting state when this method is called.

`client` - DAVIC ResourceClient to be used for resource management.

`requestData` - application specific data to be used for resource management. Null is possible.

**Throws:**

`java.lang.IllegalArgumentException` - if the serviceContext or the client is null. Note that requestData may be null.

`java.lang.IllegalStateException` - if this method is called before creating all filters in this VBIFilterGroup.

`org.davic.mpeg.sections.FilterResourceException` - if reserving the specified filters fails

#### detach

`public void` **`detach`**`()`

Release all filters in this VBIFilterGroup and terminate the connection to the ServiceContext. All filtering by filters in this group terminates immediately.

**Throws:**

`java.lang.IllegalStateException` - if this method is called when the filters have already been released and this method does nothing.

#### getServiceContext

`public javax.tv.service.selection.ServiceContext` **`getServiceContext`**`()`

Returns a ServiceContext instance specified by a `attach(javax.tv.service.selection.ServiceContext, org.davic.resources.ResourceClient, java.lang.Object)` method.

**Returns:**

a ServiceContext instance specified by a `attach(javax.tv.service.selection.ServiceContext, org.davic.resources.ResourceClient, java.lang.Object)` method. Null if no ServiceContext is attached at the moment.

#### getSeparatedFilteringCapability

`public boolean` **`getSeparatedFilteringCapability`**`(int[] lineNumber, int dataFormat)`

Returns true if field separated filtering of the specified VBI line numbers with the specified data format is supported by this VBIFilterGroup. At a minimum, either this method or the `getMixedFilteringCapability(int[], int)` method shall return true for the line 21 with CC1, CC2, CC3, CC4, T1, T2, T3, T4 and XDS format.

**Parameters:**

`lineNumber` - an array of line numbers used to determine filtering capability.

`dataFormat` - a data format used to determine filtering capability.

**Returns:**

true if the host supports field separated filtering of the specified line numbers with the specified data format (i.e., the newVBIFilter() method accepts FIELD_1 or FIELD_2). Otherwise returns false.

#### getMixedFilteringCapability

`public boolean` **`getMixedFilteringCapability`**`(int[] lineNumber, int dataFormat)`

Returns true if field mixed filtering of the specified VBI line numbers with the specified data format is supported by this VBIFilterGroup. At a minimum, either this method or the `getSeparatedFilteringCapability(int[], int)` method shall return true for the line 21 with CC1, CC2, CC3, CC4, T1, T2, T3, T4 and XDS format.

**Parameters:**

`lineNumber` - a line number used to determine filtering capability.

`dataFormat` - a data format used to determine filtering capability.

**Returns:**

true if the host supports field mixed filtering of the specified line numbers with the specified data format (i.e., the newVBIFilter() method accepts FIELD_MIXED). Otherwise false returns.

#### getSCTE20Capability

`public boolean` **`getSCTE20Capability`**`()`

Indicates if the host supports line 21 VBI data retrieval from user_data in MPEG picture headers as defined in ANSI/SCTE 20. See also a `newVBIFilter(int[], int, int, int, int)` method.

**Returns:**

true if the host can retrieve line 21 data from user_data in MPEG picture headers as defined in ANSI/SCTE 20.

#### getSCTE21Capability

`public boolean` **`getSCTE21Capability`**`()`

Indicates if the host supports line 21 VBI data retrieval from user_data in MPEG picture headers as defined in ANSI/SCTE 21. See also a `newVBIFilter(int[], int, int, int, int)` method.

**Returns:**

true if the host can retrieve line 21 data from user_data in MPEG picture headers as defined in ANSI/SCTE 21.

#### getClient

`public org.davic.resources.ResourceClient` **`getClient`**`()`

Returns the ResourceClient object specified in the attach() method.

**Specified by:**

`getClient` in interface `org.davic.resources.ResourceProxy`

**Returns:**

the ResourceClient object specified in the last call of the `attach(javax.tv.service.selection.ServiceContext, org.davic.resources.ResourceClient, java.lang.Object)` method. If the attach() method has never been called or the `detach()` method is called to cancel an attach call, this method returns null.

#### addResourceStatusEventListener

`public void`
**`addResourceStatusEventListener`**`(org.davic.resources.ResourceStatusListener listener)`

Add a listener object to be notified of changes in the status of resources related to VBI filters.

**Specified by:**

`addResourceStatusEventListener` in interface `org.davic.resources.ResourceServer`

**Parameters:**

`listener` - an object implementing the ResourceStatusListener interface. Multiple calls with a same listener instance is simply ignored with throwing no exception.

**Throws:**

`java.lang.IllegalArgumentException` - if the listener is null.

**removeResourceStatusEventListener**

```
public void
removeResourceStatusEventListener(org.davic.resources.ResourceStatusListener l
istener)
```

Remove a specified listener object from this VBIFilterGroup.

**Specified by:**

`removeResourceStatusEventListener` in interface `org.davic.resources.ResourceServer`

**Parameters:**

`listener` - an object implementing the ResourceStatusListener interface. This method does nothing if the specified ccListener is null, not previously added or already removed.

**Throws:**

`java.lang.IllegalArgumentException` - if the listener is null.

**org.ocap.media**

## Interface VBIFilterListener

```
public interface VBIFilterListener
```

This interface represents a VBI filter event listener. When a specific event happens, the `filterUpdate(org.ocap.media.VBIFilterEvent)` method is called with an event that has a proper event code to indicate the event.

# Method Summary

| | |
|---|---|
| void | **filterUpdate**(VBIFilterEvent event)<br>This method is called by the OCAP implementation to notify an application that a VBI event has occurred. |

# Method Detail

### filterUpdate

```
void filterUpdate(VBIFilterEvent event)
```

This method is called by the OCAP implementation to notify an application that a VBI event has occurred.

**Parameters:**

`event` - a VBIFilterEvent instance that contains a event code.

# Annex T OCAP 1.1 SI Access API

Service Information APIs provide an application with a mechanism for finding out information about available services in an interactive broadcast environment. Service Selection APIs provide an interface for selecting these services for presentation.

This section of the OCAP 1.1 Specification specifies the APIs for both Service Information and Service Selection.

## T.1 DVB-GEM and DVB_MHP Specification Correspondence

This section corresponds to Annex M of the [DVB-MHP1.1.2], which is not included in [DVB-GEM 1.1]. OCAP 1.1 does not use any of the APIs specified in this section of the [DVB-MHP1.1.2].

***Table T–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1] and [DVB-MHP1.1.2]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance | [DVB-MHP1.1.2] Section | MHP Compliance |
|---|---|---|---|---|
| Annex T OCAP 1.1 SI Access API | Annex M | Extension | Annex M | Extension |

## T.2 OCAP 1.1 Specific Requirements

This information extends the specification requirements made to the [DVB-MHP1.1.2].

### T.2.1 Extensions to DVB-MHP (Normative)

The following information is normative for the OCAP 1.1 Profile. It is specific to OCAP 1.1 and is not contained within the [DVB-MHP1.1.2].

The Service Information used in systems supporting OCAP 1.1 is more closely aligned with that used in SCTE and ATSC than in DVB. Consequently, OCAP 1.1 will follow those SI models. The OCAP SI model adopts the JavaTV APIs for SI and provides some additional APIs necessary to support the specifics of the System Information delivered to an Open Cable system. The JavaTV APIs are sufficiently abstract to support the variations of Service Information including DVB, and OCAP 1.1. The additional OCAP SI APIs are necessary to address features unique to the Open Cable environment. For example DVB is more service centric, whereas in the US, it is more branded by channel. To preserve the legacy channel branding a new concept of virtual channel Table (VCT) has been defined. Using VCT, any logical channel number can be mapped to anywhere in the physical channel space. The DVB standard on SI does not use VCT.

#### *T.2.1.1 Overview of OCAP 1.1 SI Packages*

The following packages are contained in multiple specifications including the [DVB-MHP1.1.2], and the OCAP 1.1 Profile:

- javax.tv.locator
- javax.tv.service
- javax.tv.service.guide
- javax.tv.service.navigation
- javax.tv.service.selection

- javax.tv.service.transport

The following package is defined in OCAP 1.1:

- org.ocap.si

#### *T.2.1.2 Considerations Specific to Service Information*

One of the areas of significant variation among International Standards for Digital Television is the format and structure of Service Information, collectively known as SI. The standard used for delivery of service information on the out-of-band channel in North American Cable is defined in [SCTE 65], and contains six profiles:

- Profile 1-Baseline
- Profile 2-Revision Detection
- Profile 3-Parental Advisory
- Profile 4-Standard Electronic Program Guide Data
- Profile 5-Combination
- Profile 6-PSIP Only

Any given cable system can choose carry one of these six SI profiles, which provides the primary source of service information, in case the CableCARD module is present within the OCAP receiver. When the CableCARD module is present, all applications (both service bound and unbound), SHALL use the service information delivered on the out-of-band channel.

Limited service information is also provided in the in-band channel and this is used as the primary source of service information for service bound applications in case the CableCARD module is absent in the receiver. When the CableCARD device is absent, any unbound applications that run, SHALL not have access to service information in the out-of-band channel, as defined in [SCTE 65].

As detailed above, the OCAP 1.1 SI access APIs incorporate the JavaTV Abstract SI APIs. The following sections provide a mapping from this set of APIs to the service information provided in an Open Cable environment. This mapping is defined for both the CableCARD device present and CableCARD device absent cases and describes variations associated with the different [SCTE 65] profiles.

##### *T.2.1.2.1 JavaTV Abstract SI API Considerations Specific to OCAP 1.1*

[Java TV] has defined a set of generic packages for dealing with Service Information and Service Selection. These packages have been created such that they can work with various SI standards such as DVB, ATSC, and [SCTE 65]. As such, there are some specific attributes or properties of SI elements that are unique to each specific SI standard or that are difficult to map to a common attribute. Consequently, it is important to understand how the [SCTE 65] SI profiles, map to the JavaTV Abstract SI APIs.

Of particular relevance to OCAP is the relationship between JavaTV SI and OCAP OOB constructs, as mapped by the various OOB OCAP URL forms. Specific expectations regarding JavaTV API behavior are noted in the Table below. In general, one should expect undefined behavior when calling, say, a JavaTV getService() method against an Object that cannot possibly map to a true service. Similarly, undefined behavior will result from attempts to select a service using an OOB OcapLocator.

***Table T–2 - Behavior of JavaTV getService()/select() methods with regard to OOB OcapLocators and/or OOB SI***

| JavaTV SI Method | Suggested Behavior |
|---|---|
| javax.tv.media.MediaSelectControl | The implementation SHOULD throw InvalidLocatorException if either of the select(Locator) or select(Locator []) methods is called with an OOB OcapLocator. |
| javax.tv.service.selection.ServiceContext | The implementation SHOULD throw InvalidLocatorException if the select(Locator []) method is called with an OOB OcapLocator argument. |
| javax.tv.service.SIManager | The implementation SHOULD throw InvalidLocatorException if the getService(Locator), retrieveProgramEvent(Locator, SIRequestor), retrieveServiceDetails(Locator, SIRequestor), or retrieveSIElement(Locator, SIRequestor) method is called with an OOB OcapLocator argument. |

### T.2.2 Mapping of the JavaTV SI API to Open Cable SI

A number of the fields specified in the Tables are encoded in a Multiple String Structure (MSS). Where a MSS is used in a mapping to the JavaTV APIs, the implementation SHALL extract the string component associated with the user's preferred language. In the case that there is no component for the preferred language the first string component is extracted.

The source_name() data element in the Source Name Sub-Table is encoded as a Multilingual Text String. The implementation SHALL extract a single component from this MTS in an implementation defined manner and return this component.

#### *T.2.2.1 Sources of Table Information*

##### *T.2.2.1.1 Rating Region Tables with CableCARD device present*

With CableCARD device present the content rating information is provided in the Rating Region Tables defined in [SCTE 65] which is required for all regions except Rating Region 0x01 for Profiles 3 through 6. For Rating Region 0x01 the Table information is defined in EIA-766 and SHALL be provided by the implementation. For Rating Regions other than 0x01 no Rating Region information is provided in Profiles 1 and 2.

##### *T.2.2.1.2 Rating Region Tables with CableCARD device absent*

With CableCARD device absent the content rating information is provided in the Rating Region Tables defined in PSIP and is available to bound applications from the system information delivered with the associated transport stream. No Rating Region information is available to unbound applications.

##### *T.2.2.1.3 Network Text Tables with CableCARD device present*

With CableCARD device present the service name is provided in the Source Name Sub-Table of the Network Text Table defined in [SCTE 65]. This sub-Table is required in Profiles 4 and 5 and optional in Profiles 1 through 3. If this optional sub-Table is not present then the service name information is not available.

For Profile 6 the service name is provided in the Longform Virtual Channel Table.

##### *T.2.2.1.4 Network Text Tables with CableCARD device absent*

There is no Network Text Table, the service name is provided in the Cable Virtual Channel Table.

#### *T.2.2.1.5 Short-form Virtual Channel Table with CableCARD device present*

With CableCARD device present the service type and channel number are provided in optional descriptors within the Shortform Virtual Channel Table (SVCT) defined in [SCTE 65]. If the optional descriptor is not present, then the service type and channel number information is not available.

For Profile 6 the information is provided in the Longform Virtual Channel Table.

#### *T.2.2.1.6 Short-form Virtual Channel Table with CableCARD device absent*

There is no Shortform Virtual Channel Table, since the service name is provided in the Cable Virtual Channel Table.

#### *T.2.2.1.7 Long-form Virtual Channel Table with CableCARD device present*

For Profile 6, with CableCARD device present, the service name, service type, and channel number are provided in the Longform Virtual Channel Table (LVCT), defined in [SCTE 65]. The extended channel name is provided in the optional Extended Channel Name descriptor. If this descriptor is not present, the service name is used in its place.

For Profiles 1 to 5 the information is provided in other Tables.

#### *T.2.2.1.8 Cable Virtual Channel Table with CableCARD device absent*

The service name, service type and channel number are provided in the Cable Virtual Channel Table (CVCT)defined in PSIP. The extended channel name is provided in the Extended Channel Name descriptor. This Table is only provided when the CableCARD device is absent.

#### *T.2.2.1.9 Event Information Tables with CableCARD device present*

For Profiles 4 through 6, with CableCARD device present the event information is provided in the Aggregate Event Information Tables (AEIT) defined in [SCTE 65]. For other profiles and when the CableCARD device is not present, the event information from the AEIT is not present, and any event retrieval request from the AEIT will result in a call to the notifyFailure() method of the javax.tv.service.SIRequestor interface, with a SIRequestFailureType of DATA_UNAVAILABLE.

### *T.2.2.2 javax.tv.service.RatingDimension*

The RatingDimension interface represents an individual content rating scheme against which program events are rated. The data source for this interface is the Rating Region Tables.

#### *T.2.2.2.1 getDimensionName*

Returns the dimension_name_text() associated with the specified dimension. The returned string is extracted from the dimension_name_text() according to the rules for MSS data extraction.

#### *T.2.2.2.2 getNumberofLevels*

Returns the values_defined() associated with the specified dimension.

#### *T.2.2.2.3 getRatingLevelDescription*

Returns extracted data from the abbrev_rating_value_text() in the first element and from the rating_value_text() in the second element of the string array. The returned strings are extracted according to the rules for MSS extraction.

### *T.2.2.3 javax.tv.service.Service*

The Service interface represents an abstract view on what is generally referred to as a television "service" or "channel". The data source for this interface is the Virtual Channel Table and the Network Text Table. An object implementing the Service interface shall also implement both javax.tv.service.ServiceNumber and javax.tv.service.ServiceMinorNumber.

#### *T.2.2.3.1 getName*

When the LVCT or the CVCT is provided the short_name is returned. When the LVCT is not provided but the Source Name Sub-Table is provided a component of the source_name is returned from the MTS string. When none of these Tables or sub-Tables are provided the following behavior SHALL be adhered to:

- If the Service.getServiceType method returns OCAP_ABSTRACT_SERVICE the Service.getName method SHALL return the service_id in String form.
- If the Service.getServiceType method returns s service type other than OCAP_ABSTRACT_SERVICE the Service.getName method SHALL return the source_id in String form.
- When service_id or source_id is returned from the Service.getName method the String SHALL be in the same format that would be produced by the java.lang.Integer.toHexString method.

#### *T.2.2.3.2 getServiceType*

When the LVCT or the CVCT is provided the service_type is mapped as per Table T–3. When the LVCT is not provided but the Channel Properties Descriptor is provided in the SVCT the service_type is mapped as per Table T–3. When none of these Tables or sub-Tables are provided, UNKNOWN is returned.

***Table T–3 - getServiceType Mapping***

| Service Type Code | Javax.tv.Service.ServiceType |
|---|---|
| 0x01 | ANALOG_TV |
| 0x02 | DIGITAL_TV |
| 0x03 | DIGITAL_RADIO |
| 0x04 | DATA_BROADCAST |

### *T.2.2.4 javax.tv.service.ServiceMinorNumber*

This interface extends the basic ServiceNumber interface to provide the minor number of two-part service numbers described in major.minor format.

The data source for this interface is the Virtual Channel Table.

#### *T.2.2.4.1 getMinorNumber*

When the LVCT or the CVCT is provided, and specifies a two-part channel number, the minor_channel_number is returned. When the LVCT or the CVCT is provided, and specifies a one-part channel number, -1 is returned. When the LVCT and CVCT are not provided, but the optional Two Part Channel Number descriptor is provided in the SVCT, the minor_channel_number is returned. When the LVCT and CVCT are not provided, and the option Two Part Channel Number descriptor is not provided in the SVCT, -1 is returned.

### *T.2.2.5 javax.tv.service.ServiceNumber*

This interface is used to identify services by service (or channel) numbers.

The data source for this interface is the Virtual Channel Table.

#### *T.2.2.5.1 getServiceNumber*

When the LVCT or the CVCT is provided, and specifies a two-part channel number, the major_channel_number is returned. When the LVCT or the CVCT is provided, and specifies a one-part channel number, the one_part_number is returned. When the LVCT and CVCT are not provided, but the optional Two Part Channel Number descriptor is provided in the SVCT, the major_channel_number is returned. When the LVCT and CVCT are not provided, and the optional Two Part Channel Number descriptor is not provided in the SVCT, the virtual_channel_number from the SVCT is returned. In the case of an Abstract Service a -1 is always returned.

### *T.2.2.6 javax.tv.service.SIElement*

The base interface of elements provided by the SI database.

#### *T.2.2.6.1 getServiceInformationType*

This method always returns SCTE_SI.

### *T.2.2.7 javax.tv.service.SIManager*

An SIManager represents a managing entity which has knowledge of all the broadcast resources available to a receiver.

#### *T.2.2.7.1 setPreferredLanguage*

This method sets the user preferred language in the form of an ISO 639 language code to an SIManager instance. This method shall not modify the system default preference of org.dvb.user.UserPreferenceManager.

#### *T.2.2.7.2 getPreferredLanguage*

This method returns the user preferred language for an SIManager instance, that was set by the setPreferredLanguage() method.

#### *T.2.2.7.3 getRatingDimension*

Returns the RatingDimension object corresponding to the requested dimension or throws an SIException if the requested dimension is not in the list of supported dimensions.

#### *T.2.2.7.4 getSupportedDimensions*

When Rating Region information is provided, returns an array of strings extracted from the dimension_name_text() elements in the RRT. One string is extracted from each instance of dimension_name_text() in the RRT according to the rules for MSS extraction.

When Rating Region information is not provided, returns a zero length array.

#### *T.2.2.7.5 getTransports*

This method returns an object that implements Transport, NetworkCollection and TransportStreamCollection in the javax.tv.service.transport package.

#### *T.2.2.7.6 retrieveProgramEvent*

ProgramEvent object retrieved by this method corresponds to AEIT information. See also Annex T.2.2.9.

#### *T.2.2.7.7 getService*

This method returns a Service instance based on a Locator reference. Given the proper Locator, any Service instance contained in the JavaTV SI database can be returned, including references to hidden channels regardless of the hide_guide field value. The hidden field and hide_guide field are included in the L-VCT and C-VCT. The hidden field is included in the S-VCT, and when this VCT is used the hide_guide field is defined in the channel_properties_descriptor when used; see [SCTE 65] table 5-20 and table A.2.

### *T.2.2.8 javax.tv.service.guide.ContentRatingAdvisory*

ContentRatingAdvisory indicates, for a given program event, ratings for any or all of the rating dimensions defined in the content rating system for the local rating region. The ContentRatingAdvisory interface is only available when the program event in the AEIT includes a content advisory descriptor (See javax.tv.service.guide.ProgramEvent.getRating, located in the [Java TV]).

#### *T.2.2.8.1 getDimensionNames*

Returns an array of strings from the Rating Region information with one element for each rated_dimension_j in the associated content advisory descriptor. The strings returned are extracted from the dimension_name_text() element in the RRT, corresponding to each value of rated_dimension_j. Extraction rules for MSS are applied to each dimension_name_text element. An empty string is returned for any element that is not found in the RRT.

#### *T.2.2.8.2 etDisplayText*

Returns a string extracted from the rating_description_text() MSS formatted string in the associated content advisory descriptor.

#### *T.2.2.8.3 getRatingLevel*

Returns the rating_level from the associated content advisory descriptor for the specified rating dimension.

#### *T.2.2.8.4 getRatingLevelText*

Returns a string containing data extracted from the rating_value_text() in the RRT for the rating_value specified in the content advisory descriptor and the dimension specified in the method call. The returned string is extracted according to the rules for MSS extraction.

#### *T.2.2.8.5 exceeds()*

This method returns true if a current service is blocked by the user preferred blocking value specified by GeneralPreference according to the recommended rule of the FCC 98-36, ET Docket No. 97-206 [FCC 98-36] document. A rating value of a service is provided by V-chip data of [CEA-766] or content_advisory_descriptor in PMT and AEIT defined by [SCTE 65]. No other rating information is referred by this method.

This method simply informs an application rating information and doesn't stop the service presentation automatically. Note that V-chip is always working independently of the OCAP implementation so that the service will be stopped by V-chip module. However if the application want to block the service based on proprietary rating data in EPG, the application is responsible to stop the service. Note that the blocking result by the proprietary data may be different from the result by V-chip.

This method call doesn't invoke service selection or tuning to another transport stream (i.e., if AEIT of [SCTE 65] is not provided, this method works only for the current selected service). If AEIT is provided, this method works for all services provided by [SCTE 65] SI.

An application specifies blocking values via the org.dvb.user.GeneralPreference. The string of the blocking value shall be a Message of Table 3 Allowed Rating Messages - U.S. in [CEA-766] for US rating dimension or Symbol described in Section 6.3 of [CEA-766]. Note that a space character shall be removed from Message and Symbol. Multiple blocking value can be specified by multiple GeneralPreference instance. If the service is blocked by one of GeneralPreference instances, the exceeds() method returns true (i.e., blocked).

To keep consistency of blocking value, a native V-chip module and the OCAP implementation SHALL share the blocking value (i.e., the native V-chip module shall set own blocking value specified by own user interface to the GeneralPreference). And the application shall set own blocking value to the GeneralPreference to inform the value to the native V-chip module. The native V-chip shall refer the GeneralPreference to block contents.

#### *T.2.2.9 javax.tv.service.guide.ProgramEvent*

A Program Event represents a collection of elementary streams with a common time base, an associated start time, and an associated end time. Program Event information is obtained asynchronously and is not always available, so attempts to retrieve Program Events may fail.

The OCAP implementation SHOULD return a null reference if a calling method requests (directly or indirectly) a reference to an Object that implements the ProgramEvent interface and that is associated with an OOB service described by one or more entries in the Aggregate Event Information Table (AEIT) or Aggregate Extended Text Table (AETT).

##### *T.2.2.9.1 getDuration*

Returns the duration from the corresponding AEIT event description.

##### *T.2.2.9.2 getEndTime*

Returns the sum of the start_time and the duration from the corresponding AEIT event description.

##### *T.2.2.9.3 getName*

Returns a string extracted from the title_text() MSS string from the corresponding AEIT event description.

##### *T.2.2.9.4 getRating*

When there is a content advisory descriptor associated with this event in the AEIT, returns an interface to the data contained in that content advisory descriptor. When there is no content advisory descriptor associated with this event in the AEIT, returns NULL.

##### *T.2.2.9.5 getService*

Returns the service information corresponding to the source_id in the corresponding AEIT event description.

##### *T.2.2.9.6 getStartTime*

Returns the start_time from the corresponding AEIT event description.

##### *T.2.2.9.7 retrieveComponents*

When the event is not current the notifyFailure() method of the SIRequestor class SHALL be called with an SIRequestFailureType of DATA_UNAVAILABLE. When the event is current, the implementation SHALL attempt to retrieve the information from the current service information associated with the source_id.

#### *T.2.2.9.8 retrieveDescription*

When there is no AETT associated with this event in the AEIT the notifyFailure() method of the SIRequestor class SHALL be called with an SIRequestFailureType of DATA_UNAVAILABLE. When there is an AETT associated with this event, the implementation SHALL attempt to retrieve the information from this Table.

### *T.2.2.10 javax.tv.service.guide.ProgramEventDescription*

This SIElement provides a textual description of a ProgramEvent. Program Event Description information is obtained asynchronously and is not always available, so attempts to retrieve Program Event Descriptions may fail.

#### *T.2.2.10.1 getProgramEventDescription*

Returns a string extracted from the extended_text_message() MSS string from the corresponding AETT event description.

### *T.2.2.11 javax.tv.service.guide.ProgramScheduleEvent*

A ProgramScheduleEvent notifies an ProgramScheduleListener of changes to program events detected in a ProgramSchedule. The ProgramSceduleListener is called on each change of information in the AEIT. When no AEIT information is provided, no ProgramEvent or ProgramSchedule information is available and the corresponding interface methods cannot be called.

### *T.2.2.12 javax.tv.service.guide.ProgramSchedule*

This interface represents a collection of program events for a given service ordered by time.

#### *T.2.2.12.1 addListener*

At least ProgramScheduleEvent with SIChangeType.MODIFY shall be guaranteed to be generated if one of Profile 4 to 6 of [SCTE 65] is transmitted. The other types of this event are not guaranteed to be generated as described in the method description.

### *T.2.2.13 javax.tv.service.navigation.CAIdentification*

This interface associates information related to the conditional access (CA) subsystem with certain SI objects. The values returned from the interface methods are implementation specific and interoperable applications SHOULD NOT make calls to these methods.

### *T.2.2.14 javax.tv.service.navigation.ServiceComponent*

This interface provides information about components of a current service. Instances of ServiceComponent are only available for current services (see Annex T.2.2.9.7). The details of service components are obtained from the Program Map Table (PMT).

The OCAP implementation SHOULD return a null reference if a calling method requests (directly or indirectly) a reference to an Object that implements the ServiceComponent interface and that is associated with a Program Information loop descriptor elementary stream contained in a PMT retrieved via OOB.

#### *T.2.2.14.1 getAssociatedLanguage*

Returns the three-character language code in ISO 639 format contained in the ISO_639_language_descriptor in the PMT for this service component.

#### *T.2.2.14.2 getName*

If the PMT contains a component name descriptor then this method returns a string extracted from the component_name_string() MSS string in this descriptor. Otherwise, this method returns the generic name associated with this stream type as returned by the javax.tv.navigation.StreamType.toString() method for this components stream type.

#### *T.2.2.14.3 getService*

Returns the service to which this service component belongs.

#### *T.2.2.14.4 getStreamType*

Returns the stream type as described in the following Table for mapping from the stream_type in the PMT to Java fields.

***Table T–4 - getStreamType Mapping***

| Stream type value | Java TV Stream Type |
|---|---|
| 0x00 | UNKNOWN |
| 0x01 | VIDEO |
| 0x02 | VIDEO |
| 0x03 | AUDIO |
| 0x04 | AUDIO |
| 0x05 | SECTIONS |
| 0x06 | DATA |
| 0x07 | UNKNOWN |
| 0x08 | DATA |
| 0x09 | UNKNOWN |
| 0x0A-0x0D | DATA |
| 0x0E-0x7F | UNKNOWN |
| 0x80 | VIDEO |
| 0X81 | AUDIO |
| 0x82-0xFF | UNKNOWN |

### *T.2.2.15 javax.tv.service.navigation.ServiceDescription*

This interface provides extended channel name information.

#### *T.2.2.15.1 getServiceDescription*

This method returns a string extracted from the long_channel_name_text() MSS string in the extended channel name descriptor.

### *T.2.2.16 javax.tv.service.navigation.ServiceDetails*

This interface provides access to meta-data of a specific instance of a service.

The OCAP implementation SHOULD return a null reference if a calling method requests (directly or indirectly) a reference to an Object that implements the ServiceDetails interface and is associated with an OOB service defined in a Virtual Channel Table (VCT).

#### *T.2.2.16.1 getDeliverySystemType*

This method always returns a DeliverySystemType of CABLE for both an inband service and an abstract service.

#### *T.2.2.16.2 getLongName*

For an inband service, when the Source Name Sub-Table is provided a component of the source_name is returned from the MTS string. When none of this sub-Table is not provided, NULL is returned.

For an abstract service, a service name in an abstract_service_descriptor in XAIT is returned.

#### *T.2.2.16.3 getProgramSchedule*

For an inband service, when the AEIT information is provided, this call returns the schedule of events that are associated with this service. When the AEIT is not available, NULL is returned.

This method call is synchronous, so only those events in currently stored AEIT are returned.

For an abstract service, null is returned always.

#### *T.2.2.16.4 getServiceType*

For an in-band service, when the LVCT or the CVCT is provided the service_type is mapped as per Table T–1. When the LVCT is not provided but the Channel Properties Descriptor is provided in the SVCT the service_type is mapped as per Table T–1. When none of these Tables or sub-Tables are provided, UNKNOWN is returned.

For an abstract service, OCAP_ABSTRACT_SERVICE is returned.

#### *T.2.2.16.5 retrieveServiceDescription*

For an inband service, when the extended channel name descriptor is omitted from the LVCT or CVCT, this method results in a call to the notifyFailure() method of the SIRequestor class with an SIRequestFailureType of DATA_UNAVAILABLE. Otherwise, the information from the extended channel name descriptor is provided through the ServiceDescription interface.

For an abstract service, this method shall always call the notifyFailure() method of the SIRequester argument, passing a value of DATA_UNAVAILABLE.

#### *T.2.2.16.6 retrieveComponent*

For an inband service, retrieves components from the PMT associated with the program_number for this service as detailed in the LVCT or SVCT.

For an abstract service, this method shall always call the notifyFailure() method of the SIRequester argument, passing a value of DATA_UNAVAILABLE.

#### *T.2.2.16.7 addServiceComponentChangeListener*

At least ServiceComponentChangeEvent with SIChangeType.MODIFY shall be guaranteed to be generated for all Profiles of [SCTE 65]. The other types of this event are not guaranteed to be generated as described in the method description.

#### *T.2.2.17 javax.tv.service.navigation.ServiceProviderInformation*

This interface is not implemented in OCAP.

#### *T.2.2.18 javax.tv.service.transport.Bouquet*

This interface is not implemented in OCAP.

#### *T.2.2.19 javax.tv.service.transport.Network*

This interface provides descriptive information concerning a network. The getName() and getNetworkID() methods for this interface return implementation dependent information.

#### *T.2.2.20 javax.tv.service.transport.Transport*

This interface represents an individual content delivery mechanism. The object implementing the Transport interface shall also implement the interfaces NetworkCollection and TransportStreamCollection.

##### *T.2.2.20.1 getDeliverySystemType*

This method always returns a DeliverySystemType of CABLE.

##### *T.2.2.20.2 addServiceDetailsChangeListener*

At least ServiceDetailsChangeEvent with SIChangeType.MODIFY shall be guaranteed to be generated if one of Profile 2 to 6 of [SCTE 65] is transmitted. (Note that Profile 1 doesn't support version up of SI Tables.) The other types of this event are not guaranteed to be generated as described in the method description.

#### *T.2.2.21 javax.tv.service.transport.TransportStream*

This interface provides basic information about the transport stream.

##### *T.2.2.21.1 getTransportStreamID*

When the LVCT is available this method returns the channel_TSID for the corresponding channel number from the LVCT. If the LVCT is not available and the SVCT is available this method returns the channel_TSID from the channel_properties descriptor for the corresponding virtual_channel. If neither the LVCT nor the SVCT is available and the CVCT is available this method returns the channel_TSID for the corresponding virtual channel in the CVCT. If TSID is not available from any of the LVCT, SVCT, or CVCT, then this method SHALL use the PAT to obtain the TSID. If the TSID is not available from any source, then this method SHALL return 0xFFFF.

##### *T.2.2.21.2 getDescription*

When available this method returns the short_name from the LVCT or the CVCT. If none of these sources of information is available returns an empty string.

#### *T.2.2.22 javax.tv.service.transport.NetworkCollection*

This interface represents a collection of networks on a Transport.

#### *T.2.2.22.1 addNetworkChangeListener*

At least NetworkChangeEvent with SIChangeType.MODIFY shall be guaranteed to be generated if one of Profile 2 to 6 of [SCTE 65] is transmitted. (Note that Profile 1 doesn't support version up of SI Tables.) The other types of this event are not guaranteed to be generated as described in the method description.

### *T.2.2.23 javax.tv.service.transport.TransportStreamCollection*

This interface represents a collection of transport stream on a Transport.

#### *T.2.2.23.1 addTransportStreamChangeListener*

At least TransportStreamChangeEvent with SIChangeType.MODIFY shall be guaranteed to be generated if one of Profile 2 to 6 of [SCTE 65] is transmitted. (Note that Profile 1 doesn't support version up of SI Tables.) The other types of this event are not guaranteed to be generated as described in the method description.

## T.3 SI APIs

This section presents the org.ocap.si and org.ocap.si.descriptor packages.

### Package org.ocap.si

| Interface Summary | |
|---|---|
| **DescriptorTag** | The DescriptorTag contains constant values to be read from the descriptor_tag field in a Descriptor. |
| **PATProgram** | This interface represents an program block in the information loop of the MPEG-2 PAT. Each PAT will contain a loop of these blocks. |
| **PMTElementaryStreamInfo** | This interface represents an MPEG-2 PMT Elementary Stream Info loop. |
| **ProgramAssociationTable** | This interface represents an MPEG-2 Program Association Table (PAT). |
| **ProgramMapTable** | This interface represents an MPEG-2 Program Map Table (PMT). |
| **StreamType** | This interface represents valid values for the stream_type field in the PMT, and returned by the getStreamType method from an implemented object of the ProgramMapTable interface. |
| **Table** | This interface represents an MPEG-2 Program Specific Information (PSI) table structure. |
| **TableChangeListener** | This interface is implemented by an application for notification of change to a table. |

| Class Summary | |
|---|---|
| **Descriptor** | This class represents an MPEG-2 descriptor. |
| **ProgramAssociationTableManager** | The Program Association Table (PAT) manager is used to discover and listen for PATs. |
| **ProgramMapTableManager** | The Program Map Table (PMT) manager is used to discover and listen for MPEG-2 PMTs. |

**org.ocap.si**

## Class Descriptor

```
java.lang.Object
  └ org.ocap.si.Descriptor

public abstract class Descriptor
extends java.lang.Object
```

This class represents an MPEG-2 descriptor.

# Constructor Summary

| |
|---|
| **Descriptor**() |

# Method Summary

| | |
|---|---|
| abstract byte | **getByteAt**(int index)<br>Get a particular byte within the descriptor content. |
| abstract byte[] | **getContent**()<br>Get the data contained within this descriptor. |
| abstract short | **getContentLength**()<br>Get the descriptor_length field. |
| abstract short | **getTag**()<br>Get the descriptor_tag field. |

| **Methods inherited from class java.lang.Object** |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Constructor Detail

### Descriptor

```
public Descriptor()
```

# Method Detail

### getTag

```
public abstract short getTag()
```

Get the descriptor_tag field. Eight bit field that identifies each descriptor. The range of valid MPEG-2 descriptor tag values is 0x2 through 0xFF.

**Returns:**

The descriptor tag.

### getContentLength

```
public abstract short getContentLength()
```

Get the descriptor_length field. Eight bit field specifying the number of bytes of the descriptor immediately following the descriptor_length field.

**Returns:**

The descriptor length.

### getContent

```
public abstract byte[] getContent()
```

Get the data contained within this descriptor. The data is returned in an array of bytes and consists of the data immediately following the descriptor_length field with length indicated by that field.

**Returns:**

The descriptor data.

### getByteAt

```
public abstract byte getByteAt(int index)
                        throws java.lang.IndexOutOfBoundsException
```

Get a particular byte within the descriptor content. The data is returned in a byte which is located at the position specified by an index parameter in the data content immediately following the descriptor_length field.

**Parameters:**

`index` - An index to the descriptor content. Value 0 corresponds to the first byte after the length field.

**Returns:**

The required byte data.

**Throws:**

`java.lang.IndexOutOfBoundsException` - if index < 0 or index >= ContentLength

**org.ocap.si**

## Interface DescriptorTag

```
public interface DescriptorTag
```

The DescriptorTag contains constant values to be read from the descriptor_tag field in a Descriptor. The constant values will correspond to those specified in OCAP SI.

| Field Summary | |
|---|---|
| static short | **AC3_AUDIO**<br>SCTE AC3_audio_descriptor, ATSC AC-3_audio_stream_descriptor or ATSC audio_stream_descriptor. |
| static short | **APPLICATION**<br>MHP application_descriptor. |
| static short | **APPLICATION_ICONS**<br>MHP application_icons_descriptor. |
| static short | **APPLICATION_NAME**<br>MHP application_name_descriptor. |
| static short | **APPLICATION_SIGNALING**<br>MHP application_signaling_descriptor. |
| static short | **ASSOCIATION_TAG**<br>DSMCC association_tag_descriptor |
| static short | **ATSC_PRIVATE_INFORMATION**<br>ATSC ATSC_private_information_descriptor |
| static short | **AUDIO_STREAM**<br>MPEG-2 audio_stream_descriptor. |
| static short | **CA**<br>MPEG-2 CA_descriptor. |
| static short | **CACHING_PRIORITY**<br>MHP caching_priority_descriptor. |
| static short | **CAPTION_SERVICE**<br>SCTE caption_service_descriptor. |
| static short | **CAROUSEL_IDENTIFIER**<br>DSMCC carousel_identifier_descriptor |
| static short | **CHANNEL_PROPERTIES**<br>SCTE channel_properties_descriptor |
| static short | **COMPONENT_NAME**<br>SCTE component_name_descriptor. |
| static short | **CONTENT_ADVISORY**<br>SCTE content_advisory_descriptor. |
| static short | **CONTENT_TYPE**<br>MHP content_type_descriptor. |

| Field Summary | |
|---|---|
| static short | **COPYRIGHT**<br>MPEG-2 copyright_descriptor. |
| static short | **DATA_BROADCAST_ID**<br>MHP data_broadcast_id_descriptor. |
| static short | **DATA_STREAM_ALIGNMENT**<br>MPEG-2 data_stream_alignment_descriptor. |
| static short | **DAYLIGHT_SAVINGS_TIME**<br>SCTE daylight_savings_time_descriptor. |
| static short | **DII_LOCATION**<br>MHP DII_location_descriptor. |
| static short | **DVB_J_APPLICATION**<br>MHP DVB-J_application_descriptor. |
| static short | **DVB_J_APPLICATION_LOCATION**<br>MHP DVB-J_application_location_descriptor. |
| static short | **EXTENDED_CHANNEL_NAME_DESCRIPTION**<br>SCTE extended_channel_name_description_descriptor. |
| static short | **EXTERNAL_APPLICATION_AUTHORISATION**<br>MHP external_application_authorisation_descriptor. |
| static short | **HIERARCHY**<br>MPEG-2 hierarchy_descriptor. |
| static short | **IBP**<br>MPEG-2 IBP_descriptor. |
| static short | **IPV4_ROUTING**<br>MHP IPV4_routing_descriptor. |
| static short | **IPV6_ROUTING**<br>MHP IPV6_routing_descriptor. |
| static short | **ISO_639_LANGUAGE**<br>MPEG-2 ISO_639_language_descriptor. |
| static short | **LABEL**<br>MHP label_descriptor. |
| static short | **MAC_ADDRESS_LIST**<br>MAC address list |
| static short | **MAXIMUM_BITRATE**<br>MPEG-2 maximum_bitrate_descriptor. |
| static short | **MULTIPLEX_UTILIZATION_BUFFER**<br>MPEG-2 multiplex_utilization_buffer_descriptor. |
| static short | **PRE_FETCH**<br>MHP pre-fetch_descriptor. |
| static short | **PRIVATE_DATA_INDICATOR**<br>MPEG-2 private_data_indicator_descriptor. |
| static short | **PRIVATE_DATA_SPECIFIER**<br>MHP private_data_specifier_descriptor. |

# Field Summary

| | |
|---|---|
| static short | **REGISTRATION**<br>MPEG-2 registration_descriptor. |
| static short | **REVISION_DETECTION**<br>SCTE revision_detection_descriptor. |
| static short | **SERVICE_IDENTIFIER**<br>MHP service_identifier_descriptor. |
| static short | **SERVICE_LOCATION**<br>SCTE service_location_descriptor. |
| static short | **SMOOTHING_BUFFER**<br>MPEG-2 smoothing_buffer_descriptor. |
| static short | **STD**<br>MPEG-2 STD_descriptor. |
| static short | **STREAM_IDENTIFIER**<br>DVB stream_identifier_descriptor. |
| static short | **STUFFING**<br>SCTE stuffing_descriptor. |
| static short | **SYSTEM_CLOCK**<br>MPEG-2 system_clock_descriptor. |
| static short | **TARGET_BACKGROUND_GRID**<br>MPEG-2 target_background_grid_descriptor. |
| static short | **TIME_SHIFTED_SERVICE**<br>SCTE time_shifted_service_descriptor. |
| static short | **TRANSPORT_PROTOCOL**<br>MHP transport_protocol_descriptor. |
| static short | **TWO_PART_CHANNEL_NUMBER**<br>SCTE two_part_channel_number_descriptor. |
| static short | **VIDEO_STREAM**<br>MPEG-2 video_stream_descriptor. |
| static short | **VIDEO_WINDOW**<br>MPEG-2 video_window_descriptor. |

# Field Detail

### VIDEO_STREAM

```
static final short VIDEO_STREAM
```
MPEG-2 video_stream_descriptor.

### AUDIO_STREAM

```
static final short AUDIO_STREAM
```
MPEG-2 audio_stream_descriptor.

#### HIERARCHY

```
static final short HIERARCHY
```
MPEG-2 hierarchy_descriptor.

#### REGISTRATION

```
static final short REGISTRATION
```
MPEG-2 registration_descriptor.

#### DATA_STREAM_ALIGNMENT

```
static final short DATA_STREAM_ALIGNMENT
```
MPEG-2 data_stream_alignment_descriptor.

#### TARGET_BACKGROUND_GRID

```
static final short TARGET_BACKGROUND_GRID
```
MPEG-2 target_background_grid_descriptor.

#### VIDEO_WINDOW

```
static final short VIDEO_WINDOW
```
MPEG-2 video_window_descriptor.

#### CA

```
static final short CA
```
MPEG-2 CA_descriptor.

#### ISO_639_LANGUAGE

```
static final short ISO_639_LANGUAGE
```
MPEG-2 ISO_639_language_descriptor.

#### SYSTEM_CLOCK

```
static final short SYSTEM_CLOCK
```
MPEG-2 system_clock_descriptor.

#### MULTIPLEX_UTILIZATION_BUFFER

```
static final short MULTIPLEX_UTILIZATION_BUFFER
```
MPEG-2 multiplex_utilization_buffer_descriptor.

#### COPYRIGHT

```
static final short COPYRIGHT
```
MPEG-2 copyright_descriptor.

#### MAXIMUM_BITRATE

```
static final short MAXIMUM_BITRATE
```
MPEG-2 maximum_bitrate_descriptor.

#### PRIVATE_DATA_INDICATOR

```
static final short PRIVATE_DATA_INDICATOR
```
MPEG-2 private_data_indicator_descriptor.

#### SMOOTHING_BUFFER

```
static final short SMOOTHING_BUFFER
```
MPEG-2 smoothing_buffer_descriptor.

#### STD

```
static final short STD
```
MPEG-2 STD_descriptor.

#### IBP

```
static final short IBP
```
MPEG-2 IBP_descriptor.

#### CAROUSEL_IDENTIFIER

```
static final short CAROUSEL_IDENTIFIER
```
DSMCC carousel_identifier_descriptor

#### ASSOCIATION_TAG

```
static final short ASSOCIATION_TAG
```
DSMCC association_tag_descriptor

#### APPLICATION

```
static final short APPLICATION
```
MHP application_descriptor.

#### APPLICATION_NAME

```
static final short APPLICATION_NAME
```
MHP application_name_descriptor.

#### TRANSPORT_PROTOCOL

```
static final short TRANSPORT_PROTOCOL
```
MHP transport_protocol_descriptor.

#### DVB_J_APPLICATION

```
static final short DVB_J_APPLICATION
```
MHP DVB-J_application_descriptor.

#### DVB_J_APPLICATION_LOCATION

```
static final short DVB_J_APPLICATION_LOCATION
```
MHP DVB-J_application_location_descriptor.

#### EXTERNAL_APPLICATION_AUTHORISATION

```
static final short EXTERNAL_APPLICATION_AUTHORISATION
```
MHP external_application_authorisation_descriptor.

#### IPV4_ROUTING

```
static final short IPV4_ROUTING
```
MHP IPV4_routing_descriptor.

#### IPV6_ROUTING

```
static final short IPV6_ROUTING
```
MHP IPV6_routing_descriptor.

#### APPLICATION_ICONS

```
static final short APPLICATION_ICONS
```
MHP application_icons_descriptor.

#### PRE_FETCH

```
static final short PRE_FETCH
```
MHP pre-fetch_descriptor.

#### DII_LOCATION

`static final short` **`DII_LOCATION`**
MHP DII_location_descriptor.

#### STREAM_IDENTIFIER

`static final short` **`STREAM_IDENTIFIER`**
DVB stream_identifier_descriptor. Used when resolving component tags (e.g. when mounting a carousel). Defined in ETSI EN 300 468.

#### PRIVATE_DATA_SPECIFIER

`static final short` **`PRIVATE_DATA_SPECIFIER`**
MHP private_data_specifier_descriptor.

#### DATA_BROADCAST_ID

`static final short` **`DATA_BROADCAST_ID`**
MHP data_broadcast_id_descriptor.

#### APPLICATION_SIGNALING

`static final short` **`APPLICATION_SIGNALING`**
MHP application_signaling_descriptor.

#### SERVICE_IDENTIFIER

`static final short` **`SERVICE_IDENTIFIER`**
MHP service_identifier_descriptor.

#### LABEL

`static final short` **`LABEL`**
MHP label_descriptor.

#### CACHING_PRIORITY

`static final short` **`CACHING_PRIORITY`**
MHP caching_priority_descriptor.

#### CONTENT_TYPE

`static final short` **`CONTENT_TYPE`**
MHP content_type_descriptor.

#### STUFFING

`static final short` **`STUFFING`**
SCTE stuffing_descriptor.

#### AC3_AUDIO

`static final short` **`AC3_AUDIO`**
SCTE AC3_audio_descriptor, ATSC AC-3_audio_stream_descriptor or ATSC audio_stream_descriptor. All names carry the same tag value of 0x81 and have the same syntax (but have slightly differing constraints on semantics and recommend usage.

#### CAPTION_SERVICE

`static final short` **`CAPTION_SERVICE`**
SCTE caption_service_descriptor.

#### CONTENT_ADVISORY

`static final short` **`CONTENT_ADVISORY`**
SCTE content_advisory_descriptor.

#### REVISION_DETECTION

`static final short` **`REVISION_DETECTION`**
SCTE revision_detection_descriptor.

#### TWO_PART_CHANNEL_NUMBER

`static final short` **`TWO_PART_CHANNEL_NUMBER`**
SCTE two_part_channel_number_descriptor.

#### CHANNEL_PROPERTIES

`static final short` **`CHANNEL_PROPERTIES`**
SCTE channel_properties_descriptor

#### DAYLIGHT_SAVINGS_TIME

`static final short` **`DAYLIGHT_SAVINGS_TIME`**
SCTE daylight_savings_time_descriptor.

#### EXTENDED_CHANNEL_NAME_DESCRIPTION

`static final short` **`EXTENDED_CHANNEL_NAME_DESCRIPTION`**
SCTE extended_channel_name_description_descriptor.

#### SERVICE_LOCATION

`static final short` **`SERVICE_LOCATION`**
SCTE service_location_descriptor.

#### TIME_SHIFTED_SERVICE

`static final short` **`TIME_SHIFTED_SERVICE`**
SCTE time_shifted_service_descriptor.

#### COMPONENT_NAME

`static final short` **`COMPONENT_NAME`**
SCTE component_name_descriptor.

#### MAC_ADDRESS_LIST

`static final short` **`MAC_ADDRESS_LIST`**
MAC address list

#### ATSC_PRIVATE_INFORMATION

`static final short` **`ATSC_PRIVATE_INFORMATION`**
ATSC ATSC_private_information_descriptor

**org.ocap.si**
## Interface PATProgram

```
public interface PATProgram
```

This interface represents an program block in the information loop of the MPEG-2 PAT.Each PAT will contain a loop of these blocks.

# Method Summary

| | |
|---|---|
| int | **getPID**()<br>Get the program_map_PID field. |
| int | **getProgramNumber**()<br>Get the program_number field. |

# Method Detail

### getProgramNumber

```
int getProgramNumber()
```

Get the program_number field. Sixteen bit field, specifies the program to which the program_map_PID applies. If the value is 0x0000, the program_map_PID, as returned bygetPID(), references the network PID.

**Returns:**
The program number.

### getPID

```
int getPID()
```

Get the program_map_PID field. Thirteen bit field specifying the PID of the transport stream packets.

**Returns:**
The program map PID.

**org.ocap.si**
## Interface PMTElementaryStreamInfo

```
public interface PMTElementaryStreamInfo
```

This interface represents an MPEG-2 PMT Elementary Stream Info loop. Each PMT will contain a loop of these blocks, as a block per an elementary stream.

# Method Summary

| | |
|---|---|
| Descriptor[] | **getDescriptorLoop**()<br>Get the descriptors associated with the elementary stream. |
| short | **getElementaryPID**()<br>Get the elementary_PID field. |
| java.lang.String | **getLocatorString**()<br>Get the locator for the elementary stream. |
| short | **getStreamType**()<br>Get the stream_type field. |

# Method Detail

### getStreamType

```
short getStreamType()
```

Get the stream_type field. Eight bit field specifying the type of program element carried within the packets within the PID returned by getElementaryPID(). See the StreamType interface for defined values.

**Returns:**
The stream type.

### getElementaryPID

```
short getElementaryPID()
```

Get the elementary_PID field. Thirteen bit field specifying the PID of the associated elementary stream.

**Returns:**
The elementary PID.

### getDescriptorLoop

```
Descriptor[] getDescriptorLoop()
```

Get the descriptors associated with the elementary stream.

**Returns:**
The descriptor loop.

### getLocatorString

`java.lang.String` **`getLocatorString`**`()`

Get the locator for the elementary stream.

For an Inband PMT, the returned OcapLocator corresponds to one of the following OCAP URL forms:

```
ocap://source_id.@&ltcomponent_tag>{&&ltcomponent_tag>}
ocap://source_id.+PID
ocap://f=frequency.program_number.@&ltcomponent_tag>{&&ltcomponent_tag>}
ocap://f=frequency.program_number.+PID
```

The forms including the PID are returned if and only if no component tags are signaled. The form returned (apart from the component tag and PID elements) must correspond to the form of the OCAP URL passed to the previous call to `ProgramMapTableManager.retrieveInBand()` or `ProgramMapTableManager.addInBandChangeListener()`.

For an OOB PMT, the returned OcapLocator corresponds to one of the following OCAP URL forms:

```
ocap://oobfdc.program_number.@&ltcomponent_tag>{&&ltcomponent_tag>}
ocap://oobfdc.program_number.+PID
```

The form including the PID is returned if and only if no component tags are signaled.

**Returns:**

The string which represents the URL of the elementary stream represented by this PMTElementaryStreamInfo.

**org.ocap.si**

## Interface ProgramAssociationTable

**All Superinterfaces:**

javax.tv.service.SIElement, javax.tv.service.SIRetrievable, Table

```
public interface ProgramAssociationTable
extends Table
```

This interface represents an MPEG-2 Program Association Table (PAT).

For an Inband PAT, the getLocator() method defined in the SIElement interface shall return an org.ocap.net.OcapLocator instance corresponding to one of the following OCAP URL forms: `ocap://source_id` `ocap://f=frequency.program_number`
The form returned must match the form of the OCAP URL passed to the previous call to `ProgramAssociationTableManager.retrieveInBand()` or `ProgramAssociationTableManager.addInBandChangeListener()`.
For an OOB PAT, the returned OcapLocator corresponds to the following OCAP URL form: `ocap://oobfdc.program_number`
The getServiceInformationType() method defined in the SIElement interface shall return ServiceInformationType.UNKNOWN.

# Method Summary

| | |
|---|---|
| PATProgram[] | **getPrograms**()<br>Get the program loop in Program Association Table. |
| short | **getTableId**()<br>Get the identifier for this table. |

| Methods inherited from interface javax.tv.service.SIElement |
|---|
| equals, getLocator, getServiceInformationType, hashCode |

| Methods inherited from interface javax.tv.service.SIRetrievable |
|---|
| getUpdateTime |

# Method Detail

### getTableId

```
short getTableId()
```

Get the identifier for this table.

**Specified by:**

`getTableId` in interface `Table`

**Returns:**

The table identifier.

### getPrograms

`PATProgram[] getPrograms()`

Get the program loop in Program Association Table.

**Returns:**

The list of PATProgram which represents the program loop in Program Association Table.

**org.ocap.si**

## Class ProgramAssociationTableManager

```
java.lang.Object
  └ org.ocap.si.ProgramAssociationTableManager
```

```
public abstract class ProgramAssociationTableManager
extends java.lang.Object
```

The Program Association Table (PAT) manager is used to discover and listen for PATs. To retrieve the PAT, an application add the TableChangeListener to ProgramAssociationTableManager via the addInBandChangeListener() or the addOutOfBandChangeListener(), and call the retrieveInBand() or the retrieveOutOfBand(). If PAT has changed, ProgramAssociationTableManager calls the TableChangeListener.notifyChange() to notify it. The application must get the ProgramAssociationTable object via the SIChangeEvent.getSIElement() method to keep the PAT table fresh when the PAT change is notified, i.e., ProgramAssociationTable is not updated automatically.

## Constructor Summary

| | |
|---|---|
| protected | **ProgramAssociationTableManager**()<br>For Singleton behavior |

## Method Summary

| | |
|---|---|
| abstract void | **addInBandChangeListener**(TableChangeListener listener, javax.tv.locator.Locator locator)<br>Add a TableChangeListener object that will be notified when the inband PAT for the channel (transport stream) identified by the Locator parameter changes. |
| abstract void | **addOutOfBandChangeListener**(TableChangeListener listener)<br>Add a TableChangeListener object that will be notified when the out-of-band PAT changes. |
| static ProgramAssociationTableManager | **getInstance**()<br>Get an instance of the Program Association Table Manager. |
| abstract void | **removeInBandChangeListener**(TableChangeListener listener)<br>Remove the TableChangeListener object for the inband PAT. |
| abstract void | **removeOutOfBandChangeListener**(TableChangeListener listener)<br>Remove the TableChangeListener object for the OOB PAT. |
| abstract javax.tv.service.SIRequest | **retrieveInBand**(javax.tv.service.SIRequestor requestor, javax.tv.locator.Locator locator)<br>Retrieve a PAT from the in-band channel (transport stream) identified by the Locator parameter. |

# Method Summary

| | |
|---|---|
| abstract javax.tv.service.SIRequest | **retrieveOutOfBand**(javax.tv.service.SIRequestor requestor)<br>Retrieve the PAT from the out-of-band channel. |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Constructor Detail

### ProgramAssociationTableManager

```
protected ProgramAssociationTableManager()
```

For Singleton behavior

# Method Detail

### getInstance

```
public static ProgramAssociationTableManager getInstance()
```

Get an instance of the Program Association Table Manager.

**Returns:**

The ProgramAssociationTableManager instance.

### addInBandChangeListener

```
public abstract void addInBandChangeListener(TableChangeListener listener,
                                             javax.tv.locator.Locator locator)
```

Add a TableChangeListener object that will be notified when the inband PAT for the channel (transport stream) identified by the Locator parameter changes. If the specified TableChangeListener object is already added, no action is performed.

**Parameters:**

`listener` - A TableChangeListener object to be informed when the inband PAT changes.

`locator` - A locator to specify the channels (transport streams) carry the PATs. Should correspond to one of the following OCAP URL forms: ocap://source_id, ocap://f=frequency.program_number

### addOutOfBandChangeListener

```
public abstract void addOutOfBandChangeListener(TableChangeListener listener)
```

Add a TableChangeListener object that will be notified when the out-of-band PAT changes. If the specified TableChangeListener object is already added, no action is performed.

**Parameters:**

`listener` - A TableChangeListener object to be informed when the out-of-band PAT changes.

#### removeInBandChangeListener

```
public abstract void removeInBandChangeListener(TableChangeListener listener)
```

Remove the TableChangeListener object for the inband PAT.

**Parameters:**

`listener` - The TableChangeListener object to be removed.

#### removeOutOfBandChangeListener

```
public abstract void
removeOutOfBandChangeListener(TableChangeListener listener)
```

Remove the TableChangeListener object for the OOB PAT.

**Parameters:**

`listener` - The TableChangeListener object to be removed.

#### retrieveInBand

```
public abstract javax.tv.service.SIRequest
retrieveInBand(javax.tv.service.SIRequestor requestor,

javax.tv.locator.Locator locator)
```

Retrieve a PAT from the in-band channel (transport stream) identified by the Locator parameter.

The OCAP implementation does not automatically tune to the transport stream specified by the Locator. Hence, the calling application must tune to the corresponding transport stream before calling this method. The attempt to retrieve a PAT stops silently and permanently when the network interface starts tuning to another transport stream. In this case, the registered `SIRequestor.notifyFailure()` method is invoked with a failure type of `javax.tv.service.SIRequestFailureType.DATA_UNAVAILABLE`.

It is not guaranteed that the transport stream specified by the Locator is still tuned when the method of the SIRequestor is called back.

Note: If an application has added a listener via the `addInBandChangeListener()` method, the `TableChangeListener.notifyChange()` method is called when the specified PAT is updated. In this case, the PAT returned to the `SIRequestor` registered with this method may have expired.

**Parameters:**

`requestor` - The SIRequestor object to be called back with the retrieval result.

`locator` - A locator to specify the channels (transport streams) carrying the PATs. Should correspond to one of the following OCAP URL forms: ocap://source_id, ocap://f=frequency.program_number

**Returns:**

The SIRequest object that identifies this asynchronous retrieval request and allows the request to be cancelled.

#### retrieveOutOfBand

```
public abstract javax.tv.service.SIRequest
retrieveOutOfBand(javax.tv.service.SIRequestor requestor)
```

Retrieve the PAT from the out-of-band channel. If there is no OOB PAT or the OCAP implementation does not support retrieving a PAT via OOB, the SIRequestor.notifyFailure method will be called with a failure type of javax.tv.service.SIRequestFailureType.DATA_UNAVAILABLE.

**Parameters:**

`requestor` - The SIRequestor object to be called back with the retrieval result.

**Returns:**
The SIRequest object that identifies this asynchronous retrieval request and allows the request to be cancelled.

**org.ocap.si**

## Interface ProgramMapTable

**All Superinterfaces:**

javax.tv.service.SIElement, javax.tv.service.SIRetrievable, Table

```
public interface ProgramMapTable
extends Table
```

This interface represents an MPEG-2 Program Map Table (PMT).

For an Inband PMT, the getLocator() method defined in the SIElement interface shall return an org.ocap.net.OcapLocator instance corresponding to one of the following OCAP URL forms:
`ocap://source_id`
`ocap://f=frequency.program_number`
The form returned must match the form of the OCAP URL passed to the previous call to `ProgramMapTableManager.retrieveInBand()` or `ProgramMapTableManager.addInBandChangeListener()`.
For an OOB PMT, the returned OcapLocator corresponds to the following OCAP URL form:
`ocap://oobfdc.program_number`
The getServiceInformationType() method defined in the SIElement interface shall return ServiceInformationType.UNKNOWN.

## Method Summary

| | |
|---|---|
| `Descriptor[]` | **`getOuterDescriptorLoop`**`()`<br>Get the outer descriptor loop. |
| `int` | **`getPcrPID`**`()`<br>Get the PCR_PID field. |
| `PMTElementaryStreamInfo[]` | **`getPMTElementaryStreamInfoLoop`**`()`<br>Get elementary stream information blocks. |
| `int` | **`getProgramNumber`**`()`<br>Get the program_number field, corresponds with the PMT. |

| **Methods inherited from interface org.ocap.si.Table** |
|---|
| `getTableId` |

| **Methods inherited from interface javax.tv.service.SIElement** |
|---|
| `equals, getLocator, getServiceInformationType, hashCode` |

| **Methods inherited from interface javax.tv.service.SIRetrievable** |
|---|
| `getUpdateTime` |

# Method Detail

### getProgramNumber

`int` **`getProgramNumber`**`()`

Get the program_number field, corresponds with the PMT.

**Returns:**

The program number corresponds with the PMT.

### getPcrPID

`int` **`getPcrPID`**`()`

Get the PCR_PID field. Thirteen bit field indicates the PID that shall contain the PCR fields of the transport stream packets.

**Returns:**

The PCR PID.

### getOuterDescriptorLoop

`Descriptor[]` **`getOuterDescriptorLoop`**`()`

Get the outer descriptor loop. List of descriptors that pertains to all programs.

**Returns:**

The outer descriptor loop.

### getPMTElementaryStreamInfoLoop

`PMTElementaryStreamInfo[]` **`getPMTElementaryStreamInfoLoop`**`()`

Get elementary stream information blocks. Each block contains elementary stream data for a particular stream type.

**Returns:**

The elementary stream information blocks.

**org.ocap.si**
## Class ProgramMapTableManager

```
java.lang.Object
  └ org.ocap.si.ProgramMapTableManager
```

```
public abstract class ProgramMapTableManager
extends java.lang.Object
```

The Program Map Table (PMT) manager is used to discover and listen for MPEG-2 PMTs. To retrieve the PMT, an application add the TableChangeListener to the ProgramMapTableManager via the addInBandChangeListener() or the addOutOfBandChangeListener(), and call the retrieveInBand() or the retrieveOutOfBand(). If PMT has changed, ProgramMapTableManager call TableChangeListener.NotifyChange() to notify it. The application must get updated ProgramMapTable object via the SIChangeEvent.getSIElement() to keep the PMT table fresh when the PMT change is notified, i.e., ProgramMapTable object is not updated automatically.

## Constructor Summary

| | |
|---|---|
| protected | **ProgramMapTableManager**()<br>For Singleton behavior |

## Method Summary

| | |
|---|---|
| abstract void | **addInBandChangeListener**(TableChangeListener listener, javax.tv.locator.Locator locator)<br>Add a TableChangeListener object that will be notified when the inband PMT changes. |
| abstract void | **addOutOfBandChangeListener**(TableChangeListener listener, int programNumber)<br>Add a TableChangeListener object that will be notified when the out-of-band PMT changes. |
| static ProgramMapTableManager | **getInstance**()<br>Get an instance of the Program Map Table Manager. |
| abstract void | **removeInBandChangeListener**(TableChangeListener listener)<br>Remove the TableChangedListener object for the inband PMT. |
| abstract void | **removeOutOfBandChangeListener**(TableChangeListener listener)<br>Remove the TableChangedListener object for the OOB PMT. |
| abstract javax.tv.service.SIRequest | **retrieveInBand**(javax.tv.service.SIRequestor requestor, javax.tv.locator.Locator locator)<br>Retrieve a PMT from the in-band channel (transport stream) identified by the Locator parameter. |
| abstract javax.tv.service.SIRequest | **retrieveOutOfBand**(javax.tv.service.SIRequestor requestor, int programNumber)<br>Retrieve the PMT from the out-of-band channel. |

| Methods inherited from class java.lang.Object |
| --- |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Constructor Detail

### ProgramMapTableManager

```
protected ProgramMapTableManager()
```

For Singleton behavior

# Method Detail

### getInstance

```
public static ProgramMapTableManager getInstance()
```

Get an instance of the Program Map Table Manager.

**Returns:**

The ProgramMapTableManager instance.

### addInBandChangeListener

```
public abstract void addInBandChangeListener(TableChangeListener listener,
                                             javax.tv.locator.Locator locator)
```

Add a TableChangeListener object that will be notified when the inband PMT changes. If the specified TableChangeListener object is already added, no action is performed.

**Parameters:**

`listener` - A TableChangeListener object to be notified when the inband PMT changes.

`locator` - A locator to specify a virtual channel carrying the inband PMTs. Should correspond to one of the following OCAP URL forms: ocap://source_id, ocap://f=frequency.program_number

### addOutOfBandChangeListener

```
public abstract void addOutOfBandChangeListener(TableChangeListener listener,
                                                int programNumber)
```

Add a TableChangeListener object that will be notified when the out-of-band PMT changes. If the specified TableChangeListener object is already added, no action is performed.

**Parameters:**

`listener` - A TableChangeListener object to be notified when the out-of-band PMT changes.

`programNumber` - A program number of the PMT from the corresponding PAT.

### removeInBandChangeListener

```
public abstract void removeInBandChangeListener(TableChangeListener listener)
```

Remove the TableChangedListener object for the inband PMT.

**Parameters:**

`listener` - The TableChangeListener object to be removed.

#### removeOutOfBandChangeListener

```
public abstract void
removeOutOfBandChangeListener(TableChangeListener listener)
```

Remove the TableChangedListener object for the OOB PMT.

**Parameters:**

`listener` - The TableChangeListener object to be removed.

#### retrieveInBand

```
public abstract javax.tv.service.SIRequest
retrieveInBand(javax.tv.service.SIRequestor requestor,

javax.tv.locator.Locator locator)
```

Retrieve a PMT from the in-band channel (transport stream) identified by the Locator parameter.

The OCAP implementation does not automatically tune to the transport stream specified by the Locator. Hence, the calling application must tune to the corresponding transport stream before calling this method. The attempt to retrieve a PMT stops silently and permanently when the network interface starts tuning to another transport stream. In this case, the registered `SIRequestor.notifyFailure()` method is invoked with a failure type of `javax.tv.service.SIRequestFailureType.DATA_UNAVAILABLE`.

It is not guaranteed that the transport stream specified by the Locator is still tuned when the method of the SIRequestor is called back.

Note: If an application has added a listener via the `addInBandChangeListener()` method, the `TableChangeListener.notifyChange()` method is called when the specified PMT is updated. In this case, the PMT returned to the `SIRequestor` registered with this method may have expired.

**Parameters:**

`requestor` - The SIRequestor object to be called back with the retrieval result, when the PMT is discovered.

`locator` - A locator to specify a virtual channel carrying the inband PMTs. Should correspond to one of the following OCAP URL forms: ocap://source_id, ocap://f=frequency.program_number

**Returns:**

The SIRequest object that identifies this asynchronous retrieval request and allows the request to be cancelled.

#### retrieveOutOfBand

```
public abstract javax.tv.service.SIRequest
retrieveOutOfBand(javax.tv.service.SIRequestor requestor,

int programNumber)
```

Retrieve the PMT from the out-of-band channel. If there is no OOB PMT or the OCAP implementation does not support retrieving a PMT via OOB, the SIRequestor.notifyFailure method will be called with a failure type of javax.tv.service.SIRequestFailureType.DATA_UNAVAILABLE.

**Parameters:**

`requestor` - The SIRequestor object to be called back with the retrieval result.

`programNumber` - A program number of the PMT from the corresponding PAT.

**Returns:**

The SIRequest object that identifies this asynchronous retrieval request and allows the request to be cancelled.

**org.ocap.si**

## Interface StreamType

```
public interface StreamType
```

This interface represents valid values for the stream_type field in the PMT, and returned by the getStreamType method from an implemented object of the ProgramMapTable interface.

| Field Summary | |
|---|---|
| static short | **ASYNCHRONOUS_DATA**<br>Asynchronous data [SCTE 53]. |
| static short | **ATSC_AUDIO**<br>ATSC A/53 audio [ATSC A/53E] |
| static short | **AUXILIARY**<br>ISO/IEC 13818-1 [ISO 13818-1] auxiliary. |
| static short | **DSM_CC**<br>13818-1 [ISO 13818-1] Annex A - DSM CC. |
| static short | **DSM_CC_MPE**<br>ISO/IEC 13818-6 [ISO 13818-6] type A (Multi-protocol Encapsulation). |
| static short | **DSM_CC_SECTIONS**<br>ISO/IEC 13818-6 [ISO 13818-6] type D (DSM-CC Sections any type, including private data). |
| static short | **DSM_CC_STREAM_DESCRIPTORS**<br>ISO/IEC 13818-6 [ISO 13818-6] type C (DSM-CC Stream Descriptors). |
| static short | **DSM_CC_UN**<br>ISO/IEC 13818-6 [ISO 13818-6] type B (DSM-CC U-N Messages). |
| static short | **H_222**<br>ITU-T Rec. |
| static short | **ISOCHRONOUS_DATA**<br>Isochronous data [SCTE 19]. |
| static short | **MHEG**<br>ISO/IEC 13818-1 [ISO 13818-1] MHEG. |
| static short | **MPEG_1_AUDIO**<br>ISO/IEC 11172-3 [ISO 11172-3 Audio. |
| static short | **MPEG_1_VIDEO**<br>ISO/IEC 11172-2 [ISO 11172-2] Video. |
| static short | **MPEG_2_AUDIO**<br>ISO/IEC 13818-3 [ISO 13818-3] Audio. |
| static short | **MPEG_2_VIDEO**<br>ITU-T Rec.H.222.1. |
| static short | **MPEG_PRIVATE_DATA**<br>ITU-T Rec. |

# Field Summary

| | |
|---|---|
| `static short` | **`MPEG_PRIVATE_SECTION`**<br>ITU-T Rec. |
| `static short` | **`STD_SUBTITLE`**<br>Standard subtitle. |
| `static short` | **`VIDEO_DCII`**<br>DigiCipher II video. |

# Field Detail

### ASYNCHRONOUS_DATA

`static final short` **`ASYNCHRONOUS_DATA`**
Asynchronous data [SCTE 53].

### ATSC_AUDIO

`static final short` **`ATSC_AUDIO`**
ATSC A/53 audio [ATSC A/53E]

### AUXILIARY

`static final short` **`AUXILIARY`**
ISO/IEC 13818-1 [ISO 13818-1] auxiliary.

### DSM_CC

`static final short` **`DSM_CC`**
13818-1 [ISO 13818-1] Annex A - DSM CC.

### DSM_CC_MPE

`static final short` **`DSM_CC_MPE`**
ISO/IEC 13818-6 [ISO 13818-6] type A (Multi-protocol Encapsulation).

### DSM_CC_SECTIONS

`static final short` **`DSM_CC_SECTIONS`**
ISO/IEC 13818-6 [ISO 13818-6] type D (DSM-CC Sections any type, including private data).

### DSM_CC_STREAM_DESCRIPTORS

`static final short` **`DSM_CC_STREAM_DESCRIPTORS`**
ISO/IEC 13818-6 [ISO 13818-6] type C (DSM-CC Stream Descriptors).

#### DSM_CC_UN

```
static final short DSM_CC_UN
```
ISO/IEC 13818-6 [ISO 13818-6] type B (DSM-CC U-N Messages).

#### H_222

```
static final short H_222
```
ITU-T Rec. H.222.1.

#### ISOCHRONOUS_DATA

```
static final short ISOCHRONOUS_DATA
```
Isochronous data [SCTE 19].

#### MHEG

```
static final short MHEG
```
ISO/IEC 13818-1 [ISO 13818-1] MHEG.

#### MPEG_1_AUDIO

```
static final short MPEG_1_AUDIO
```
ISO/IEC 11172-3 [ISO 11172-3 Audio.

#### MPEG_1_VIDEO

```
static final short MPEG_1_VIDEO
```
ISO/IEC 11172-2 [ISO 11172-2] Video.

#### MPEG_2_AUDIO

```
static final short MPEG_2_AUDIO
```
ISO/IEC 13818-3 [ISO 13818-3] Audio.

#### MPEG_2_VIDEO

```
static final short MPEG_2_VIDEO
```
ITU-T Rec.H.222.1.

#### MPEG_PRIVATE_DATA

```
static final short MPEG_PRIVATE_DATA
```
ITU-T Rec. H.222.1.

#### MPEG_PRIVATE_SECTION

`static final short` **`MPEG_PRIVATE_SECTION`**
ITU-T Rec. H.222.1.

#### STD_SUBTITLE

`static final short` **`STD_SUBTITLE`**
Standard subtitle.

#### VIDEO_DCII

`static final short` **`VIDEO_DCII`**
DigiCipher II video.

**org.ocap.si**

## Interface Table

**All Superinterfaces:**
javax.tv.service.SIElement, javax.tv.service.SIRetrievable

**All Known Subinterfaces:**
ProgramAssociationTable, ProgramMapTable

```
public interface Table
extends javax.tv.service.SIElement
```

This interface represents an MPEG-2 Program Specific Information (PSI) table structure.

# Method Summary

| | |
|---|---|
| short | **getTableId**()<br>Returns the table_id field of the table. |

| Methods inherited from interface javax.tv.service.SIElement |
|---|
| equals, getLocator, getServiceInformationType, hashCode |

| Methods inherited from interface javax.tv.service.SIRetrievable |
|---|
| getUpdateTime |

# Method Detail

### getTableId

```
short getTableId()
```

Returns the table_id field of the table. Eight bit field that identifies the table.

**Returns:**
The table Id.

**org.ocap.si**

## Interface TableChangeListener

**All Superinterfaces:**

java.util.EventListener, javax.tv.service.SIChangeListener

```
public interface TableChangeListener
extends javax.tv.service.SIChangeListener
```

This interface is implemented by an application for notification of change to a table. The ProgramAassociationTableManager and the ProgramMapTableManager call the TableChangeListener.notifyChange method with the concrete event sub class of the org.javax.tv.service.SIChangeEvent. The concrete event sub class of the SIChangeEvent depends on the implement.

# Method Summary

| | |
|---|---|
| void | **notifyChange**(javax.tv.service.SIChangeEvent event)<br>This method notifies that the SI table has changed. |

# Method Detail

### notifyChange

```
void notifyChange(javax.tv.service.SIChangeEvent event)
```

This method notifies that the SI table has changed.

**Parameters:**

`event` - An event instance that notifies the SI update.

# Annex U OCAP 1.1 System Event API

The system event API allows an application with MonitorAppPermission("systemevent") to register for and receive error, resource depletion, and reboot events; see Section 21.2.1.19 for a description.

***Table U–1 - Correlation between OCAP 1.1 and [DVB-GEM 1.1]***

| OCAP | [DVB-GEM 1.1] Section | GEM Compliance |
|---|---|---|
| Annex U OCAP 1.1 System Event API | No Corresponding Section | OCAP-Specific Extension |

Following is a Java example demonstrating how an application can register to be the error event handler:

```
public class EventListenerAppSample implements SystemEventListener
{
    private final static int MAX_EVENT_STORE = 5;
    private static int eventCount = 0;
    private SystemEvent[] imeStore = new SystemEvent[MAX_EVENT_STORE];

    /**
     * The zero argument constructor demonstrates a possible application example where
     * the application registers to receive error events.
     */
    public EventListenerAppSample()
    {
        // Get the system event manager.
        SystemEventManager sem = SystemEventManager.getInstance();

        // Set this object as the new error event listener.
        sem.setEventListener(SystemEventManager.ERROR_EVENT_LISTENER, this);
    }

    /**
     * Receives a message event from the implementation. This method will be used to process
     * all of the error and informational messages sent to this registered error listener.
     *
     * This sample simply places the messages into an array. Additional processing is
     * specific to the application. For example, an application may look at the error code
     * and application identifier of the event and take recovery action for specific errors,
     * in which case it would return null. The application may return non-null indicating that
     * it has changed the event.
     *
     * @param see Event generated by the system or sent by an application.
     */
    public void notifyEvent(SystemEvent me)
    {
        System.out.print("ErrorListenerAppSample.notifyEvent(); event type: ");
        System.out.print(me.getTypeCode());
        System.out.print(";  date: ");
        System.out.println(me.getDate());

        eventCount = (eventCount == MAX_EVENT_STORE - 1) ? 0 : eventCount + 1;

        imeStore[eventCount] = me;     // Store the event for later retrieval.
    }
}
```

Following is a Java example demonstrating how an application can log an error:

```
import org.ocap.event.*;

public class EventSenderSample
{
/** Our application-specific error code. */
    private static final int ID_FOR_APP_SAMPLE = SystemEvent.BEGIN_APP_REC_ERROR_TYPES + 42;

    public static void sendTestErrorEvent() {
```

```
        // Create an error event.
        ErrorEvent ee = new ErrorEvent(ID_FOR_APP_SAMPLE, "TestEvent");

        // Get the default system event logger.
        SystemEventManager sem = SystemEventManager.getInstance();

        // Log an error to the default system error handler.
        sem.log(ee);
    }
}
```

### Package org.ocap.system.event

| Interface Summary | |
|---|---|
| **SystemEventListener** | System event handler implemented by a trusted application and registered with `SystemEventManager`. |

| Class Summary | |
|---|---|
| **DeferredDownloadEvent** | This class represents an event returned by the system when a deferred download is instigated by the receipt of a code version table with download_command = 0x01 (see [CCIF 2.0]). |
| **ErrorEvent** | This class represents an event returned by the system when an uncaught exception or implementation error is encountered, or by an application that wishes to log an error or an informational event. |
| **RebootEvent** | This class represents an event returned by the system when a reboot is instigated. |
| **ResourceDepletionEvent** | Event that indicates resources are low, and the system is about to destroy application(s) to attempt to correct this. |
| **SystemEvent** | This class is the basis for system event messages. |
| **SystemEventManager** | Registration mechanism for trusted applications to set the error handler. |

#### org.ocap.system.event
## Class DeferredDownloadEvent

```
java.lang.Object
  └ org.ocap.system.event.SystemEvent
      └ org.ocap.system.event.DeferredDownloadEvent
```

```
public class DeferredDownloadEvent
extends SystemEvent
```

This class represents an event returned by the system when a deferred download is instigated by the receipt of a code version table with download_command = 0x01 (see [CCIF 2.0].

# Field Summary

| Fields inherited from class org.ocap.system.event.SystemEvent |
| --- |
| BEGIN_APP_CAT_ERROR_EVENT_TYPES, BEGIN_APP_CAT_ERROR_RESERVED_EVENT_TYPES, BEGIN_APP_INFO_EVENT_TYPES, BEGIN_APP_INFO_RESERVED_EVENT_TYPES, BEGIN_APP_REC_ERROR_EVENT_TYPES, BEGIN_APP_REC_ERROR_RESERVED_EVENT_TYPES, BEGIN_SYS_CAT_ERROR_EVENT_TYPES, BEGIN_SYS_CAT_ERROR_RESERVED_EVENT_TYPES, BEGIN_SYS_DNLD_EVENT_TYPES, BEGIN_SYS_INFO_EVENT_TYPES, BEGIN_SYS_INFO_RESERVED_EVENT_TYPES, BEGIN_SYS_REBOOT_EVENT_TYPES, BEGIN_SYS_REBOOT_RESERVED_EVENT_TYPES, BEGIN_SYS_REC_ERROR_EVENT_TYPES, BEGIN_SYS_REC_ERROR_RESERVED_EVENT_TYPES, BEGIN_SYS_RES_DEP_EVENT_TYPES, BEGIN_SYS_RES_DEP_RESERVED_EVENT_TYPES, END_APP_CAT_ERROR_EVENT_TYPES, END_APP_INFO_EVENT_TYPES, END_APP_REC_ERROR_EVENT_TYPES, END_SYS_CAT_ERROR_EVENT_TYPES, END_SYS_DNLD_EVENT_TYPES, END_SYS_INFO_EVENT_TYPES, END_SYS_REBOOT_EVENT_TYPES, END_SYS_REC_ERROR_EVENT_TYPES, END_SYS_RES_DEP_EVENT_TYPES |

# Constructor Summary

| |
| --- |
| **DeferredDownloadEvent**(int typeCode)<br>System event constructor assigns an eventId, Date, and ApplicationIdentifier. |

# Method Summary

| Methods inherited from class org.ocap.system.event.SystemEvent |
| --- |
| getAppID, getDate, getMessage, getTypeCode |

| Methods inherited from class java.lang.Object |
| --- |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Constructor Detail

### DeferredDownloadEvent

```
public DeferredDownloadEvent(int typeCode)
                      throws java.lang.IllegalArgumentException
```

System event constructor assigns an eventId, Date, and ApplicationIdentifier.

**Parameters:**

`typeCode` - - Unique event type.

**Throws:**

`java.lang.IllegalArgumentException` - if the typeCode is not in a defined application range when the event is created by an application.

**org.ocap.system.event**

## Class ErrorEvent

```
java.lang.Object
  └ org.ocap.system.event.SystemEvent
      └ org.ocap.system.event.ErrorEvent

public class ErrorEvent
extends SystemEvent
```

This class represents an event returned by the system when an uncaught exception or implementation error is encountered, or by an application that wishes to log an error or an informational event. Error event type codes are defined in this class. Applications may use the error type codes in this class or proprietary class codes that are understood by the network.

The class takes a java.lang.Throwable object parameter in one constructor, but the Throwable object cannot be returned from this class due to implementation and security issues. The Throwable object attributes (i.e., message and stacktrace) can be retrieved from this class by calling corresponding get methods, which in-turn call the Throwable object get methods. However, the implementation MUST NOT allow the Throwable object get methods to block indefinitely when called and MUST NOT wait longer than 30 seconds for them to return.

## Field Summary

| | |
|---|---|
| static int | **APP_CAT_GENERAL_ERROR**<br>Application catastrophic error that doesn't fit into any other given category. |
| static int | **APP_INFO_GENERAL_EVENT**<br>Application informational event that doesn't fit into any other given category. |
| static int | **APP_REC_GENERAL_ERROR**<br>Application recoverable error that doesn't fit into any other given category. |
| static int | **APP_REC_JAVA_THROWABLE**<br>Application recoverable error - a Java Throwable caught by an exception, but that can be recovered from by the application, or a Throwable that was created by an application due to detection of a recoverable event. |
| static int | **SYS_CAT_GENERAL_ERROR**<br>System catastrophic error that doesn't fit into any other given category. |
| static int | **SYS_CAT_JAVA_THROWABLE**<br>Java Throwable thrown by a call made by an application but not caught by the application. |
| static int | **SYS_INFO_GENERAL_EVENT**<br>System informational event that doesn't fit into any other given category. |
| static int | **SYS_REC_GENERAL_ERROR**<br>System error that doesn't fit into any other given category. |

| Fields inherited from class org.ocap.system.event.SystemEvent |
|---|
| BEGIN_APP_CAT_ERROR_EVENT_TYPES, BEGIN_APP_CAT_ERROR_RESERVED_EVENT_TYPES, BEGIN_APP_INFO_EVENT_TYPES, BEGIN_APP_INFO_RESERVED_EVENT_TYPES, BEGIN_APP_REC_ERROR_EVENT_TYPES, BEGIN_APP_REC_ERROR_RESERVED_EVENT_TYPES, BEGIN_SYS_CAT_ERROR_EVENT_TYPES, BEGIN_SYS_CAT_ERROR_RESERVED_EVENT_TYPES, BEGIN_SYS_DNLD_EVENT_TYPES, BEGIN_SYS_INFO_EVENT_TYPES, BEGIN_SYS_INFO_RESERVED_EVENT_TYPES, BEGIN_SYS_REBOOT_EVENT_TYPES, BEGIN_SYS_REBOOT_RESERVED_EVENT_TYPES, BEGIN_SYS_REC_ERROR_EVENT_TYPES, BEGIN_SYS_REC_ERROR_RESERVED_EVENT_TYPES, BEGIN_SYS_RES_DEP_EVENT_TYPES, BEGIN_SYS_RES_DEP_RESERVED_EVENT_TYPES, END_APP_CAT_ERROR_EVENT_TYPES, END_APP_INFO_EVENT_TYPES, END_APP_REC_ERROR_EVENT_TYPES, END_SYS_CAT_ERROR_EVENT_TYPES, END_SYS_DNLD_EVENT_TYPES, END_SYS_INFO_EVENT_TYPES, END_SYS_REBOOT_EVENT_TYPES, END_SYS_REC_ERROR_EVENT_TYPES, END_SYS_RES_DEP_EVENT_TYPES |

# Constructor Summary

| |
|---|
| **ErrorEvent**(int typeCode, java.lang.String message)<br>Class constructor specifying the event type code and readable message. |
| **ErrorEvent**(int typeCode, java.lang.String message, java.lang.String stacktrace, java.lang.String[] throwableClasses, long date, org.dvb.application.AppID appId)<br>This constructor is provided for internal use by OCAP implementations; applications SHOULD NOT call it. |
| **ErrorEvent**(int typeCode, java.lang.Throwable throwable)<br>Class constructor specifying the event type code, and throwable condition. |

# Method Summary

| | |
|---|---|
| java.lang.String | **getMessage**()<br>Gets the readable message String that was passed to a constructor explicitly or within a Throwable object. |
| java.lang.String | **getStackTrace**()<br>Gets the stack trace from the Throwable object if a Throwable object was passed to the appropriate constructor. |
| java.lang.String[] | **getThrowableClasses**()<br>Gets the class hierarchy from the Throwable object that was passed to the corresponding constructor. |

| Methods inherited from class org.ocap.system.event.SystemEvent |
|---|
| getAppID, getDate, getTypeCode |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Field Detail

### APP_INFO_GENERAL_EVENT

`public static final int` **`APP_INFO_GENERAL_EVENT`**

Application informational event that doesn't fit into any other given category.

### APP_REC_GENERAL_ERROR

`public static final int` **`APP_REC_GENERAL_ERROR`**

Application recoverable error that doesn't fit into any other given category.

### APP_REC_JAVA_THROWABLE

`public static final int` **`APP_REC_JAVA_THROWABLE`**

Application recoverable error - a Java Throwable caught by an exception, but that can be recovered from by the application, or a Throwable that was created by an application due to detection of a recoverable event.

### APP_CAT_GENERAL_ERROR

`public static final int` **`APP_CAT_GENERAL_ERROR`**

Application catastrophic error that doesn't fit into any other given category.

### SYS_INFO_GENERAL_EVENT

`public static final int` **`SYS_INFO_GENERAL_EVENT`**

System informational event that doesn't fit into any other given category.

### SYS_REC_GENERAL_ERROR

`public static final int` **`SYS_REC_GENERAL_ERROR`**

System error that doesn't fit into any other given category.

### SYS_CAT_GENERAL_ERROR

`public static final int` **`SYS_CAT_GENERAL_ERROR`**

System catastrophic error that doesn't fit into any other given category.

### SYS_CAT_JAVA_THROWABLE

`public static final int` **`SYS_CAT_JAVA_THROWABLE`**

Java Throwable thrown by a call made by an application but not caught by the application. This event is generated by the implementation, but indicates that an application cannot continue normal operations.

# Constructor Detail

### ErrorEvent

```
public ErrorEvent(int typeCode,
                  java.lang.String message)
           throws java.lang.IllegalArgumentException
```

Class constructor specifying the event type code and readable message.

**Parameters:**

`typeCode` - - Unique error type code.

`message` - - Readable error message.

**Throws:**

`java.lang.IllegalArgumentException` - when called by an application and the typeCode is not in one of the following ranges: `SystemEvent.BEGIN_APP_INFO_EVENT_TYPES` to `SystemEvent.END_APP_INFO_EVENT_TYPES`, or `SystemEvent.BEGIN_APP_REC_ERROR_EVENT_TYPES` to `SystemEvent.END_APP_REC_ERROR_EVENT_TYPES`, or `SystemEvent.BEGIN_APP_CAT_ERROR_EVENT_TYPES` to `SystemEvent.END_APP_CAT_ERROR_EVENT_TYPES`.

### ErrorEvent

```
public ErrorEvent(int typeCode,
                  java.lang.Throwable throwable)
           throws java.lang.IllegalArgumentException
```

Class constructor specifying the event type code, and throwable condition. The message is derived from the Throwable object. The

**Parameters:**

`typeCode` - - The unique error type code.

`throwable` - - A throwable object that was generated by the implementation or an application in response to an informational or error event, or by the implementation when a call made by an application throws an exception that isn't caught by the application.

**Throws:**

`java.lang.IllegalArgumentException` - when called by an application and the typeCode is not in one of the following ranges: `SystemEvent.BEGIN_APP_INFO_EVENT_TYPES` to `SystemEvent.END_APP_INFO_EVENT_TYPES`, or `SystemEvent.BEGIN_APP_REC_ERROR_EVENT_TYPES` to `SystemEvent.END_APP_REC_ERROR_EVENT_TYPES`, or `SystemEvent.BEGIN_APP_CAT_ERROR_EVENT_TYPES` to `SystemEvent.END_APP_CAT_ERROR_EVENT_TYPES`.

### ErrorEvent

```
public ErrorEvent(int typeCode,
                  java.lang.String message,
                  java.lang.String stacktrace,
                  java.lang.String[] throwableClasses,
                  long date,
                  org.dvb.application.AppID appId)
```

This constructor is provided for internal use by OCAP implementations; applications SHOULD NOT call it.

**Parameters:**

`typeCode` - - The unique error type code.
`message` - - Readable message specific to the event generator.
`stacktrace` - - Stacktrace taken from a Throwable object or null if no Throwable used.
`throwableClasses` - - The class hierarchy list from a Throwable object or null if no Throwable used.
`date` - - Event date in milli-seconds from midnight January 1, 1970 GMT.
`appId` - - The Id of the application logging the event.
**Throws:**
`java.lang.SecurityException` - if this constructor is called by any application.

# Method Detail

### getStackTrace

`public java.lang.String` **`getStackTrace`**`()`

Gets the stack trace from the Throwable object if a Throwable object was passed to the appropriate constructor.
**Returns:**
The stack trace from the Throwable object, or null if no Throwable object is available, or if the message cannot be extracted from the Throwable object (perhaps Throwable.printStackTrace() threw an exception or blocked).

### getMessage

`public java.lang.String` **`getMessage`**`()`

Gets the readable message String that was passed to a constructor explicitly or within a Throwable object.
**Overrides:**
`getMessage` in class `SystemEvent`
**Returns:**
The readable message, if the message cannot be extracted from the Throwable object (perhaps Throwable.getMessage() threw an exception or blocked).

### getThrowableClasses

`public java.lang.String[]` **`getThrowableClasses`**`()`

Gets the class hierarchy from the Throwable object that was passed to the corresponding constructor. Each String in the array will be a fully qualified class name. The first will be the full class name (with package name) of the Throwable object passed to this class. Each subsequent String shall contain the name of a super class up to but not including java.lang.Object.
**Returns:**
The stack trace from the Throwable object, or null if no Throwable object is available.

**org.ocap.system.event**
## Class RebootEvent

```
java.lang.Object
  └ org.ocap.system.event.SystemEvent
      └ org.ocap.system.event.RebootEvent
```

```
public class RebootEvent
extends SystemEvent
```

This class represents an event returned by the system when a reboot is instigated. Reboot event type codes are defined in this class. Implementations may use the reboot type codes in this class or proprietary class codes that are understood by the network.

# Field Summary

| | |
|---|---|
| static int | **REBOOT_BY_IMPLEMENTATION**<br>Reboot instigated by implementation; no error encountered. |
| static int | **REBOOT_BY_TRUSTED_APP**<br>Reboot instigated by trusted application. |
| static int | **REBOOT_FOR_UNRECOVERABLE_HW_ERROR**<br>Reboot instigated by the implementation; unrecoverable hardware error encountered. |
| static int | **REBOOT_FOR_UNRECOVERABLE_SYS_ERROR**<br>Reboot instigated by implementation; unrecoverable system error encountered. |

| **Fields inherited from class org.ocap.system.event.SystemEvent** |
|---|
| BEGIN_APP_CAT_ERROR_EVENT_TYPES, BEGIN_APP_CAT_ERROR_RESERVED_EVENT_TYPES, BEGIN_APP_INFO_EVENT_TYPES, BEGIN_APP_INFO_RESERVED_EVENT_TYPES, BEGIN_APP_REC_ERROR_EVENT_TYPES, BEGIN_APP_REC_ERROR_RESERVED_EVENT_TYPES, BEGIN_SYS_CAT_ERROR_EVENT_TYPES, BEGIN_SYS_CAT_ERROR_RESERVED_EVENT_TYPES, BEGIN_SYS_DNLD_EVENT_TYPES, BEGIN_SYS_INFO_EVENT_TYPES, BEGIN_SYS_INFO_RESERVED_EVENT_TYPES, BEGIN_SYS_REBOOT_EVENT_TYPES, BEGIN_SYS_REBOOT_RESERVED_EVENT_TYPES, BEGIN_SYS_REC_ERROR_EVENT_TYPES, BEGIN_SYS_REC_ERROR_RESERVED_EVENT_TYPES, BEGIN_SYS_RES_DEP_EVENT_TYPES, BEGIN_SYS_RES_DEP_RESERVED_EVENT_TYPES, END_APP_CAT_ERROR_EVENT_TYPES, END_APP_INFO_EVENT_TYPES, END_APP_REC_ERROR_EVENT_TYPES, END_SYS_CAT_ERROR_EVENT_TYPES, END_SYS_DNLD_EVENT_TYPES, END_SYS_INFO_EVENT_TYPES, END_SYS_REBOOT_EVENT_TYPES, END_SYS_REC_ERROR_EVENT_TYPES, END_SYS_RES_DEP_EVENT_TYPES |

# Constructor Summary

| |
|---|
| **RebootEvent**(int typeCode, java.lang.String message)<br>System event constructor assigns an eventId, Date, and ApplicationIdentifier. |

# Constructor Summary

**RebootEvent**(int typeCode, java.lang.String message, long date, org.dvb.application.AppID appId)
This constructor is provided for internal use by OCAP implementations; applications SHOULD NOT call it.

# Method Summary

| Methods inherited from class org.ocap.system.event.SystemEvent |
| --- |
| getAppID, getDate, getMessage, getTypeCode |

| Methods inherited from class java.lang.Object |
| --- |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Field Detail

### REBOOT_BY_IMPLEMENTATION

public static final int **REBOOT_BY_IMPLEMENTATION**
Reboot instigated by implementation; no error encountered.

### REBOOT_FOR_UNRECOVERABLE_SYS_ERROR

public static final int **REBOOT_FOR_UNRECOVERABLE_SYS_ERROR**
Reboot instigated by implementation; unrecoverable system error encountered.

### REBOOT_FOR_UNRECOVERABLE_HW_ERROR

public static final int **REBOOT_FOR_UNRECOVERABLE_HW_ERROR**
Reboot instigated by the implementation; unrecoverable hardware error encountered. For hardware errors only, not firmware. If indistinguishable between software or firmware errors in certain implementations, REBOOT_FOR_UNRECOVERABLE_SYS_ERROR MUST be generated instead.

### REBOOT_BY_TRUSTED_APP

public static final int **REBOOT_BY_TRUSTED_APP**
Reboot instigated by trusted application.
**See Also:**
Host.reboot(), Constant Field Values

# Constructor Detail

### RebootEvent

```
public RebootEvent(int typeCode,
                   java.lang.String message)
            throws java.lang.IllegalArgumentException
```

System event constructor assigns an eventId, Date, and ApplicationIdentifier.

**Parameters:**

`typeCode` - - Unique event type.

`message` - - Readable message specific to the event generator.

**Throws:**

`java.lang.IllegalArgumentException` - if the typeCode is not in a defined application range when the event is created by an application.

### RebootEvent

```
public RebootEvent(int typeCode,
                   java.lang.String message,
                   long date,
                   org.dvb.application.AppID appId)
```

This constructor is provided for internal use by OCAP implementations; applications SHOULD NOT call it.

**Parameters:**

`typeCode` - - The unique error type code.

`message` - - Readable message specific to the event generator.

`date` - - Event date in milli-seconds from midnight January 1, 1970 GMT.

`appId` - - The Id of the application logging the event.

**Throws:**

`java.lang.SecurityException` - if this constructor is called by any application.

**org.ocap.system.event**

## Class ResourceDepletionEvent

```
java.lang.Object
  └ org.ocap.system.event.SystemEvent
      └ org.ocap.system.event.ResourceDepletionEvent
```

```
public class ResourceDepletionEvent
extends SystemEvent
```

Event that indicates resources are low, and the system is about to destroy application(s) to attempt to correct this.

# Field Summary

| | |
|---|---|
| static int | **RESOURCE_CPU_BANDWIDTH_DEPLETED**<br>Available CPU cycles is depleted to an implementation specific threshold. |
| static int | **RESOURCE_RC_BANDWIDTH_DEPLETED**<br>Available reverse channel bandwidth is depleted to implementation specific threshold. |
| static int | **RESOURCE_SYS_MEM_DEPLETED**<br>Overall system memory is depleted to an implementation specific threshold. |
| static int | **RESOURCE_VM_MEM_DEPLETED**<br>VM memory for an application is depleted to an implementation specific threshold. |

| **Fields inherited from class org.ocap.system.event.SystemEvent** |
|---|
| BEGIN_APP_CAT_ERROR_EVENT_TYPES, BEGIN_APP_CAT_ERROR_RESERVED_EVENT_TYPES, BEGIN_APP_INFO_EVENT_TYPES, BEGIN_APP_INFO_RESERVED_EVENT_TYPES, BEGIN_APP_REC_ERROR_EVENT_TYPES, BEGIN_APP_REC_ERROR_RESERVED_EVENT_TYPES, BEGIN_SYS_CAT_ERROR_EVENT_TYPES, BEGIN_SYS_CAT_ERROR_RESERVED_EVENT_TYPES, BEGIN_SYS_DNLD_EVENT_TYPES, BEGIN_SYS_INFO_EVENT_TYPES, BEGIN_SYS_INFO_RESERVED_EVENT_TYPES, BEGIN_SYS_REBOOT_EVENT_TYPES, BEGIN_SYS_REBOOT_RESERVED_EVENT_TYPES, BEGIN_SYS_REC_ERROR_EVENT_TYPES, BEGIN_SYS_REC_ERROR_RESERVED_EVENT_TYPES, BEGIN_SYS_RES_DEP_EVENT_TYPES, BEGIN_SYS_RES_DEP_RESERVED_EVENT_TYPES, END_APP_CAT_ERROR_EVENT_TYPES, END_APP_INFO_EVENT_TYPES, END_APP_REC_ERROR_EVENT_TYPES, END_SYS_CAT_ERROR_EVENT_TYPES, END_SYS_DNLD_EVENT_TYPES, END_SYS_INFO_EVENT_TYPES, END_SYS_REBOOT_EVENT_TYPES, END_SYS_REC_ERROR_EVENT_TYPES, END_SYS_RES_DEP_EVENT_TYPES |

# Constructor Summary

| |
|---|
| **ResourceDepletionEvent**(int typeCode, java.lang.String message)<br>System event constructor assigns an eventId, Date, and ApplicationIdentifier. |
| **ResourceDepletionEvent**(int typeCode, java.lang.String message, long date, org.dvb.application.AppID appId)<br>This constructor is provided for internal use by OCAP implementations; applications SHOULD NOT call it. |

# Method Summary

| Methods inherited from class org.ocap.system.event.SystemEvent |
| --- |
| getAppID, getDate, getMessage, getTypeCode |

| Methods inherited from class java.lang.Object |
| --- |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Field Detail

### RESOURCE_SYS_MEM_DEPLETED

`public static final int` **`RESOURCE_SYS_MEM_DEPLETED`**

Overall system memory is depleted to an implementation specific threshold. This indicates that the platform's application manager is about to destroy an application to free system memory.

### RESOURCE_VM_MEM_DEPLETED

`public static final int` **`RESOURCE_VM_MEM_DEPLETED`**

VM memory for an application is depleted to an implementation specific threshold. This indicates that the platform's application manager is about to destroy an application to free VM memory.

### RESOURCE_CPU_BANDWIDTH_DEPLETED

`public static final int` **`RESOURCE_CPU_BANDWIDTH_DEPLETED`**

Available CPU cycles is depleted to an implementation specific threshold. This indicates that the platform's application manager is about to destroy an application to free CPU cycles.

Note that the presence of this event type does not imply that the platform's application manager must destroy applications if CPU usage is too high; merely that if it does then it must first send this event.

### RESOURCE_RC_BANDWIDTH_DEPLETED

`public static final int` **`RESOURCE_RC_BANDWIDTH_DEPLETED`**

Available reverse channel bandwidth is depleted to implementation specific threshold. This indicates that the platform's application manager is about to destroy an application to free return channel bandwidth.

Note that the presence of this event type does not imply that the platform's application manager must destroy applications if return channel bandwidth usage is too high; merely that if it does then it must first send this event.

# Constructor Detail

### ResourceDepletionEvent

```
public ResourceDepletionEvent(int typeCode,
                              java.lang.String message)
                       throws java.lang.IllegalArgumentException
```

System event constructor assigns an eventId, Date, and ApplicationIdentifier.

**Parameters:**

`typeCode` - - Unique event type.

`message` - - Readable message specific to the event generator.

**Throws:**

`java.lang.IllegalArgumentException` - if the typeCode is not in a defined application range when the event is created by an application. Since there are no defined application ranges for resource depletion events, this exception will always be thrown if this constructor is called by an application.

### ResourceDepletionEvent

```
public ResourceDepletionEvent(int typeCode,
                              java.lang.String message,
                              long date,
                              org.dvb.application.AppID appId)
```

This constructor is provided for internal use by OCAP implementations; applications SHOULD NOT call it.

**Parameters:**

`typeCode` - - The unique error type code.

`message` - - Readable message specific to the event generator.

`date` - - Event date in milli-seconds from midnight January 1, 1970 GMT.

`appId` - - The Id of the application logging the event.

**Throws:**

`java.lang.SecurityException` - if this constructor is called by any application.

**org.ocap.system.event**

## Class SystemEvent

```
java.lang.Object
  └ org.ocap.system.event.SystemEvent
```

**Direct Known Subclasses:**

DeferredDownloadEvent, ErrorEvent, RebootEvent, ResourceDepletionEvent

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

This class is the basis for system event messages. Applications cannot create this class directly, they must create one of the defined subclasses instead.

The event type code is defined with ranges reserved for specific types of events. Ranges defined for the implementation using "SYS" cannot be used by applications.

Ranges defined as "reserved" will be allocated by OCAP (or other future standards). Applications and OCAP implementations SHOULD NOT use these values until their meaning is standardized.

Values in a SYS range that are not reserved may be used by OCAP implementers - their meaning is implementation-dependent.

Values in an APP range that are not reserved may be used by OCAP applications - their meaning is application-dependent. The `getAppID()` method may be useful to disambiguate these codes.

Informational events can be used for any information desired. Recoverable error events do not prevent an application or the implementation from continued execution. Catastrophic events generated by or on behalf of an application indicate that the application cannot continue execution and will be a cause for self-destruction. Reboot events generated by the implementation indicate that the implementation cannot continue execution and a system generated reboot is imminent.

## Field Summary

| | |
|---|---|
| static int | **BEGIN_APP_CAT_ERROR_EVENT_TYPES**<br>Start of range reserved for application generated catastrophic error events defined by OCAP. |
| static int | **BEGIN_APP_CAT_ERROR_RESERVED_EVENT_TYPES**<br>Start of range reserved for application generated catastrophic error events defined by OCAP. |
| static int | **BEGIN_APP_INFO_EVENT_TYPES**<br>Start of range for application generated informational events. |
| static int | **BEGIN_APP_INFO_RESERVED_EVENT_TYPES**<br>Start of range reserved for application generated informational events defined by OCAP. |
| static int | **BEGIN_APP_REC_ERROR_EVENT_TYPES**<br>Start of range for application generated recoverable error events. |
| static int | **BEGIN_APP_REC_ERROR_RESERVED_EVENT_TYPES**<br>Start of range reserved for application generated recoverable error events defined by OCAP. |
| static int | **BEGIN_SYS_CAT_ERROR_EVENT_TYPES**<br>Start of range for system generated catastrophic events. |

| Field Summary | |
|---|---|
| static int | **BEGIN_SYS_CAT_ERROR_RESERVED_EVENT_TYPES**<br>Start of range reserved for system generated catastrophic error events defined by OCAP. |
| static int | **BEGIN_SYS_DNLD_EVENT_TYPES**<br>Start of range reserved for system generated deferred download events in response to a CVT signaling a deferred download. |
| static int | **BEGIN_SYS_INFO_EVENT_TYPES**<br>Start of range for system generated informational events. |
| static int | **BEGIN_SYS_INFO_RESERVED_EVENT_TYPES**<br>Start of range reserved for system generated informational events defined by OCAP. |
| static int | **BEGIN_SYS_REBOOT_EVENT_TYPES**<br>Start of range for system generated reboot events. |
| static int | **BEGIN_SYS_REBOOT_RESERVED_EVENT_TYPES**<br>Start of range reserved for system generated reboot events defined by OCAP. |
| static int | **BEGIN_SYS_REC_ERROR_EVENT_TYPES**<br>Start of range for system generated recoverable error events. |
| static int | **BEGIN_SYS_REC_ERROR_RESERVED_EVENT_TYPES**<br>Start of range reserved for system generated recoverable error events defined by OCAP. |
| static int | **BEGIN_SYS_RES_DEP_EVENT_TYPES**<br>Start of range for system generated resource depletion events. |
| static int | **BEGIN_SYS_RES_DEP_RESERVED_EVENT_TYPES**<br>Start of range reserved for system generated resource depletion events defined by OCAP. |
| static int | **END_APP_CAT_ERROR_EVENT_TYPES**<br>End of range for application generated catastrophic error events. |
| static int | **END_APP_INFO_EVENT_TYPES**<br>End of range for application generated informational events. |
| static int | **END_APP_REC_ERROR_EVENT_TYPES**<br>End of range for application generated recoverable error events. |
| static int | **END_SYS_CAT_ERROR_EVENT_TYPES**<br>End of range for system generated catastrophic error events. |
| static int | **END_SYS_DNLD_EVENT_TYPES**<br>End of range for system deferred download events. |
| static int | **END_SYS_INFO_EVENT_TYPES**<br>End of range for system generated informational events. |
| static int | **END_SYS_REBOOT_EVENT_TYPES**<br>End of range for system generated reboot events. |
| static int | **END_SYS_REC_ERROR_EVENT_TYPES**<br>End of range for system generated recoverable error events. |
| static int | **END_SYS_RES_DEP_EVENT_TYPES**<br>End of range for system generated resource depletion events. |

# Constructor Summary

| | |
|---|---|
| protected | **SystemEvent**(int typeCode)<br>System event constructor. |
| protected | **SystemEvent**(int typeCode, java.lang.String message)<br>System event constructor with message. |
| protected | **SystemEvent**(int typeCode, java.lang.String message, long date, org.dvb.application.AppID appId)<br>This constructor is provided for internal use by OCAP implementations; applications SHOULD NOT call it. |

# Method Summary

| | |
|---|---|
| org.dvb.application.AppID | **getAppID**()<br>Gets the globally unique identifier of the application logging the event. |
| long | **getDate**()<br>Gets the event date in milli-seconds from midnight January 1, 1970, GMT. |
| java.lang.String | **getMessage**()<br>Gets the readable message. |
| int | **getTypeCode**()<br>Gets the event type code. |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Field Detail

### BEGIN_SYS_INFO_EVENT_TYPES

public static final int **BEGIN_SYS_INFO_EVENT_TYPES**

Start of range for system generated informational events.

### BEGIN_SYS_INFO_RESERVED_EVENT_TYPES

public static final int **BEGIN_SYS_INFO_RESERVED_EVENT_TYPES**

Start of range reserved for system generated informational events defined by OCAP. This reserved range ends with END_SYS_INFO_EVENT_TYPES.

### END_SYS_INFO_EVENT_TYPES

public static final int **END_SYS_INFO_EVENT_TYPES**

End of range for system generated informational events.

#### BEGIN_APP_INFO_EVENT_TYPES

`public static final int` **`BEGIN_APP_INFO_EVENT_TYPES`**
Start of range for application generated informational events.

#### BEGIN_APP_INFO_RESERVED_EVENT_TYPES

`public static final int` **`BEGIN_APP_INFO_RESERVED_EVENT_TYPES`**
Start of range reserved for application generated informational events defined by OCAP. This reserved range ends with `END_APP_INFO_EVENT_TYPES`.

#### END_APP_INFO_EVENT_TYPES

`public static final int` **`END_APP_INFO_EVENT_TYPES`**
End of range for application generated informational events.

#### BEGIN_SYS_REC_ERROR_EVENT_TYPES

`public static final int` **`BEGIN_SYS_REC_ERROR_EVENT_TYPES`**
Start of range for system generated recoverable error events.

These events may refer to a specific application, which will be identified by the `getAppID()` method, or may be internal system errors, in which case `getAppID()` will return null.

#### BEGIN_SYS_REC_ERROR_RESERVED_EVENT_TYPES

`public static final int` **`BEGIN_SYS_REC_ERROR_RESERVED_EVENT_TYPES`**
Start of range reserved for system generated recoverable error events defined by OCAP. This reserved range ends with `END_SYS_REC_ERROR_EVENT_TYPES`.

#### END_SYS_REC_ERROR_EVENT_TYPES

`public static final int` **`END_SYS_REC_ERROR_EVENT_TYPES`**
End of range for system generated recoverable error events.

#### BEGIN_APP_REC_ERROR_EVENT_TYPES

`public static final int` **`BEGIN_APP_REC_ERROR_EVENT_TYPES`**
Start of range for application generated recoverable error events.

This type of error is intended to indicate that something went wrong (e.g. a data file could not be loaded or a system call failed), but the application is designed to handle the error gracefully so does not need to terminate.

#### BEGIN_APP_REC_ERROR_RESERVED_EVENT_TYPES

`public static final int` **`BEGIN_APP_REC_ERROR_RESERVED_EVENT_TYPES`**

Start of range reserved for application generated recoverable error events defined by OCAP. This reserved range ends with END_APP_REC_ERROR_EVENT_TYPES.

#### END_APP_REC_ERROR_EVENT_TYPES

`public static final int` **`END_APP_REC_ERROR_EVENT_TYPES`**
End of range for application generated recoverable error events.

#### BEGIN_SYS_CAT_ERROR_EVENT_TYPES

`public static final int` **`BEGIN_SYS_CAT_ERROR_EVENT_TYPES`**
Start of range for system generated catastrophic events.

These the events are generated by the system when it detects a catastrophic failure that will cause (or has caused) the application identified by the `getAppID()` method to be terminated.

These events may also be internal system errors, in which case `getAppID()` will return null.

#### BEGIN_SYS_CAT_ERROR_RESERVED_EVENT_TYPES

`public static final int` **`BEGIN_SYS_CAT_ERROR_RESERVED_EVENT_TYPES`**
Start of range reserved for system generated catastrophic error events defined by OCAP. This reserved range ends with END_SYS_CAT_ERROR_EVENT_TYPES.

#### END_SYS_CAT_ERROR_EVENT_TYPES

`public static final int` **`END_SYS_CAT_ERROR_EVENT_TYPES`**
End of range for system generated catastrophic error events.

#### BEGIN_APP_CAT_ERROR_EVENT_TYPES

`public static final int` **`BEGIN_APP_CAT_ERROR_EVENT_TYPES`**
Start of range reserved for application generated catastrophic error events defined by OCAP. This reserved range ends with END_APP_CAT_ERROR_EVENT_TYPES.

#### BEGIN_APP_CAT_ERROR_RESERVED_EVENT_TYPES

`public static final int` **`BEGIN_APP_CAT_ERROR_RESERVED_EVENT_TYPES`**
Start of range reserved for application generated catastrophic error events defined by OCAP. This reserved range ends with END_APP_CAT_ERROR_EVENT_TYPES.

#### END_APP_CAT_ERROR_EVENT_TYPES

`public static final int` **`END_APP_CAT_ERROR_EVENT_TYPES`**
End of range for application generated catastrophic error events.

#### BEGIN_SYS_REBOOT_EVENT_TYPES

`public static final int` **`BEGIN_SYS_REBOOT_EVENT_TYPES`**
Start of range for system generated reboot events.

**BEGIN_SYS_REBOOT_RESERVED_EVENT_TYPES**

`public static final int` **`BEGIN_SYS_REBOOT_RESERVED_EVENT_TYPES`**
Start of range reserved for system generated reboot events defined by OCAP. This reserved range ends with `END_SYS_REBOOT_EVENT_TYPES`.

**END_SYS_REBOOT_EVENT_TYPES**

`public static final int` **`END_SYS_REBOOT_EVENT_TYPES`**
End of range for system generated reboot events.

**BEGIN_SYS_RES_DEP_EVENT_TYPES**

`public static final int` **`BEGIN_SYS_RES_DEP_EVENT_TYPES`**
Start of range for system generated resource depletion events.

**BEGIN_SYS_RES_DEP_RESERVED_EVENT_TYPES**

`public static final int` **`BEGIN_SYS_RES_DEP_RESERVED_EVENT_TYPES`**
Start of range reserved for system generated resource depletion events defined by OCAP. This reserved range ends with `END_SYS_RES_DEP_EVENT_TYPES`.

**END_SYS_RES_DEP_EVENT_TYPES**

`public static final int` **`END_SYS_RES_DEP_EVENT_TYPES`**
End of range for system generated resource depletion events.

**BEGIN_SYS_DNLD_EVENT_TYPES**

`public static final int` **`BEGIN_SYS_DNLD_EVENT_TYPES`**
Start of range reserved for system generated deferred download events in response to a CVT signaling a deferred download. This reserved range ends with `END_SYS_DNLD_EVENT_TYPES`.

**END_SYS_DNLD_EVENT_TYPES**

`public static final int` **`END_SYS_DNLD_EVENT_TYPES`**
End of range for system deferred download events.

# Constructor Detail

**SystemEvent**

```
protected SystemEvent(int typeCode)
                throws java.lang.IllegalArgumentException
```

System event constructor. Assigns a date, and AppID. The readable message is set to null.
**Parameters:**
`typeCode` - - Unique event type.
**Throws:**
`java.lang.IllegalArgumentException` - if the typeCode is not in a defined application range when the event is created by an application.

### SystemEvent

```
protected SystemEvent(int typeCode,
                      java.lang.String message)
               throws java.lang.IllegalArgumentException
```

System event constructor with message. Assigns a date, and AppID.
**Parameters:**
`typeCode` - - Unique event type.
`message` - - Readable message specific to the event generator.
**Throws:**
`java.lang.IllegalArgumentException` - if the typeCode is not in a defined application range when the event is created by an application.

### SystemEvent

```
protected SystemEvent(int typeCode,
                      java.lang.String message,
                      long date,
                      org.dvb.application.AppID appId)
```

This constructor is provided for internal use by OCAP implementations; applications SHOULD NOT call it.
**Parameters:**
`typeCode` - - The unique error type code.
`message` - - Readable message specific to the event generator.
`date` - - Event date in milli-seconds from midnight January 1, 1970 GMT.
`appId` - - The Id of the application logging the event.
**Throws:**
`java.lang.SecurityException` - if this constructor is called by any application.

# Method Detail

### getAppID

```
public org.dvb.application.AppID getAppID()
```

Gets the globally unique identifier of the application logging the event.
**Returns:**
The identifier of the application, or null for events that do not have an associated application (such as system initiated reboots).

### getTypeCode

```
public int getTypeCode()
```

Gets the event type code. Identifies a code specific to the event system.
**Returns:**

type code of the event.

### getDate

`public long` **`getDate`**`()`

Gets the event date in milli-seconds from midnight January 1, 1970, GMT. The return value from this method can be passed to the java.util.Date(long) constructor.

**Returns:**

The date the event was submitted to the system.

### getMessage

`public java.lang.String` **`getMessage`**`()`

Gets the readable message.

**Returns:**

message string of the event.

**org.ocap.system.event**

## Interface SystemEventListener

```
public interface SystemEventListener
```

System event handler implemented by a trusted application and registered with `SystemEventManager`.

# Method Summary

| | |
|---|---|
| void | **notifyEvent**(SystemEvent event)<br>Receives error, resource depletion, or reboot events from the system when a trusted application has registered as the event handler for the respective event type. |

# Method Detail

### notifyEvent

```
void notifyEvent(SystemEvent event)
```

Receives error, resource depletion, or reboot events from the system when a trusted application has registered as the event handler for the respective event type.

**Parameters:**

`event` - - The event encountered by the implementation.

**org.ocap.system.event**

## Class SystemEventManager

```
java.lang.Object
  └ org.ocap.system.event.SystemEventManager
```

```
public abstract class SystemEventManager
extends java.lang.Object
```

Registration mechanism for trusted applications to set the error handler.

## Field Summary

| | |
|---|---|
| static int | **DEFERRED_DOWNLOAD_EVENT_LISTENER**<br>Identifies the deferred download event listener. |
| static int | **ERROR_EVENT_LISTENER**<br>Identifies the system error event listener. |
| static int | **REBOOT_EVENT_LISTENER**<br>Identifies the reboot event listener. |
| static int | **RESOURCE_DEPLETION_EVENT_LISTENER**<br>Identifies the system resource depletion event listener. |

## Constructor Summary

| | |
|---|---|
| protected | **SystemEventManager**()<br>This constructor must not be used by OCAP applications. |

## Method Summary

| | |
|---|---|
| static SystemEventManager | **getInstance**()<br>Gets the singleton instance of the system event manager. |
| abstract void | **log**(SystemEvent event)<br>Logs an event. |
| abstract void | **setEventListener**(int type, SystemEventListener sel)<br>Set the system event listener specified by type and the new handler. |
| abstract void | **unsetEventListener**(int type)<br>Unset the system event handler specified by type. |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Field Detail

### ERROR_EVENT_LISTENER

`public static final int` **`ERROR_EVENT_LISTENER`**

Identifies the system error event listener.

**See Also:**

`setEventListener(int, org.ocap.system.event.SystemEventListener)`, Constant Field Values

### REBOOT_EVENT_LISTENER

`public static final int` **`REBOOT_EVENT_LISTENER`**

Identifies the reboot event listener.

**See Also:**

`setEventListener(int,org.ocap.system.event.SystemEventListener)`

### RESOURCE_DEPLETION_EVENT_LISTENER

`public static final int` **`RESOURCE_DEPLETION_EVENT_LISTENER`**

Identifies the system resource depletion event listener.

**See Also:**

`setEventListener(int,org.ocap.system.event.SystemEventListener)`

### DEFERRED_DOWNLOAD_EVENT_LISTENER

`public static final int` **`DEFERRED_DOWNLOAD_EVENT_LISTENER`**

Identifies the deferred download event listener.

**See Also:**

`setEventListener(int, org.ocap.system.event.SystemEventListener)`

# Constructor Detail

### SystemEventManager

`protected` **`SystemEventManager`**`()`

This constructor must not be used by OCAP applications. It is only provided for implementers of the OCAP APIs.

# Method Detail

### getInstance

`public static SystemEventManager` **`getInstance`**`()`

Gets the singleton instance of the system event manager.

**Returns:**

The system event manager instance.

#### setEventListener

```
public abstract void setEventListener(int type,
                                      SystemEventListener sel)
                               throws java.lang.IllegalArgumentException
```

Set the system event listener specified by type and the new handler. On a successful call, any previously set SystemEventListener for the same type is discarded. By default no SystemEventListener is set for any type.

**Parameters:**

`type` - - ERROR_EVENT_LISTENER, REBOOT_EVENT_LISTENER, RESOURCE_DEPLETION_EVENT_LISTENER, or DEFERRED_DOWNLOAD_EVENT_LISTENER.

`sel` - - System event listener created by the registering application.

**Throws:**

`java.lang.SecurityException` - if the application does not have MonitorAppPermission("systemevent")

`java.lang.IllegalArgumentException` - if type is not one of ERROR_EVENT_LISTENER, REBOOT_EVENT_LISTENER, RESOURCE_DEPLETION_EVENT_LISTENER, or DEFERRED_DOWNLOAD_EVENT_LISTENER.

#### unsetEventListener

```
public abstract void unsetEventListener(int type)
```

Unset the system event handler specified by type.

**Parameters:**

`type` - - One of ERROR_EVENT_LISTENER, REBOOT_EVENT_LISTENER, RESOURCE_DEPLETION_EVENT_LISTENER, or DEFERRED_DOWNLOAD_EVENT_LISTENER

**Throws:**

`java.lang.SecurityException` - if the application does not have MonitorAppPermission("systemevent")

#### log

```
public abstract void log(SystemEvent event)
                  throws java.lang.IllegalArgumentException
```

Logs an event. Checks the instance of the event and calls the appropriate error, reboot, or resource depletion handler and passes the even to it.

**Parameters:**

`event` - - The event to log.

**Throws:**

`java.lang.IllegalArgumentException` - if the event parameter is an instance of an application defined class (i.e., applications cannot define their own subclasses of SystemEvent and use them with this method. This is due to implementation and security issues).

# Annex V OCAP 1.1 Storage API

## Package org.ocap.storage

| Interface Summary | |
|---|---|
| **AvailableStorageListener** | This interface represents a listener that can be set to listen for high water level reached in persistent storage indicated by the dvb.persistent.root property for all applications. |
| **DetachableStorageOption** | This interface represents an external device that can be detached. |
| **LogicalStorageVolume** | This interface represents a logical volume on a storage device. |
| **RemovableStorageOption** | This interface represents a storage device that can be inserted into or removed from a bay while power is applied, e.g., memory stick. |
| **StorageManagerListener** | This interface represents a listener for changes to the set of StorageProxies (`StorageProxy`). |
| **StorageOption** | This interface represents an option that is specific to a class of storage devices, e.g., detachable or DVR media-capable. |
| **StorageProxy** | This interface represents a persistent storage device. |

| Class Summary | |
|---|---|
| **ExtendedFileAccessPermissions** | This class extends `FileAccessPermissions` to let granting applications provide read and write file access to applications that have an organisation identifier different from a granting application. |
| **StorageManager** | This class represents the storage manager which keeps track of the storage devices attached to the system. |
| **StorageManagerEvent** | Event sent to a `StorageManagerListener` registered with the `StorageManager` that a `StorageProxy` was added, removed or changed state. |

### org.ocap.storage
## Interface AvailableStorageListener

```
public interface AvailableStorageListener
```

This interface represents a listener that can be set to listen for high water level reached in persistent storage indicated by the dvb.persistent.root property for all applications.

| Method Summary | |
|---|---|
| `void` | **`notifyHighWaterMarkReached`**`()`<br>Notifies the listener a high water mark has been reached in the available memory indicated by dvb.persistent.root and available to all applications. |

# Method Detail

**notifyHighWaterMarkReached**

`void` **`notifyHighWaterMarkReached`**`()`

Notifies the listener a high water mark has been reached in the available memory indicated by dvb.persistent.root and available to all applications. The high water mark was set as a parameter in a call to the `StorageManager.addAvailableStorageListener` method.

**org.ocap.storage**

## Interface DetachableStorageOption

**All Superinterfaces:**
StorageOption

```
public interface DetachableStorageOption
extends StorageOption
```

This interface represents an external device that can be detached. The methods on this interface allow a detachable device to be detached safely. In addition, when a detachable storage device is attached for the first time, its StorageProxy provides a means to initialize the device. If initialization is needed, the StorageProxy will be in one of two states: `UNSUPPORTED_FORMAT` or `UNINITIALIZED`. When the StorageProxy is in one of these two states, the initialize method must be called before the device can be used.

# Method Summary

| | |
|---|---|
| boolean | **isDetachable**()<br>Determines whether the device associated with this storage proxy is ready to be detached. |
| void | **makeDetachable**()<br>Makes the device safe to be detached. |
| void | **makeReady**()<br>Makes the device ready for use. |

# Method Detail

### isDetachable

```
boolean isDetachable()
```

Determines whether the device associated with this storage proxy is ready to be detached.

**Returns:**
Returns true when the device is currently ready to be detached, otherwise returns false.

### makeDetachable

```
void makeDetachable()
                    throws java.io.IOException
```

Makes the device safe to be detached. Calling this method has extensive impact on applications that may currently be using the associated storage device.

1. Any in progress java.io operations that are not completed throw IOExceptions.
2. The corresponding storage proxy is either removed from the database or remains with a status of `OFFLINE`. The latter indicates that the device may be brought back online. If it is removed from the database, attempts to use the storage proxy results in an IOException.

This call may block until the filesystem can be put into a consistent state.

**Throws:**

`java.lang.SecurityException` - if the calling application does not have MonitorAppPermission("storage").
`java.io.IOException` - if the system is unable to make the device safe to detach.

### makeReady

```
void makeReady()
            throws java.io.IOException
```

Makes the device ready for use. If a detachable device is connected and in the `OFFLINE` state, this method attempts to activate the device and make it available. For example, a device may be left in an OFFLINE state after it has been made ready to detach, but not actually unplugged. This method has no effect if the device is already in the `READY` state.

**Throws:**

`java.lang.SecurityException` - if the calling application does not have MonitorAppPermission("storage").
`java.io.IOException` - if the device was not in the READY or OFFLINE state when the method was called.

**org.ocap.storage**
## Class ExtendedFileAccessPermissions

```
java.lang.Object
  └ org.dvb.io.persistent.FileAccessPermissions
      └ org.ocap.storage.ExtendedFileAccessPermissions
```

```
public class ExtendedFileAccessPermissions
extends org.dvb.io.persistent.FileAccessPermissions
```

This class extends `FileAccessPermissions` to let granting applications provide read and write file access to applications that have an organization identifier different from a granting application.

# Constructor Summary

| |
|---|
| **`ExtendedFileAccessPermissions`**`(boolean readWorldAccessRight, boolean writeWorldAccessRight, boolean readOrganisationAccessRight, boolean writeOrganisationAccessRight, boolean readApplicationAccessRight, boolean writeApplicationAccessRight, int[] otherOrganisationsReadAccessRights, int[] otherOrganisationsWriteAccessRights)`<br>This constructor encodes application, application organization, and world file access permissions as a set of booleans, and other organizations file access permissions as arrays of granted organization identifiers. |

# Method Summary

| | |
|---|---|
| `int[]` | **`getReadAccessorganisationIds`**`()`<br>Gets the array of organization identifiers with read permission. |
| `int[]` | **`getWriteAccessorganisationIds`**`()`<br>Gets the array of organization identifiers with write permission. |
| `void` | **`setPermissions`**`(boolean readWorldAccessRight, boolean writeWorldAccessRight, boolean readOrganisationAccessRight, boolean writeOrganisationAccessRight, boolean readApplicationAccessRight, boolean writeApplicationAccessRight, int[] otherOrganisationsReadAccessRights, int[] otherOrganisationsWriteAcessRights)`<br>This method allows modification of the permissions on this instance of the ExtendedFileAccessPermission class. |

| **Methods inherited from class org.dvb.io.persistent.FileAccessPermissions** |
|---|
| `hasReadApplicationAccessRight, hasReadOrganisationAccessRight, hasReadWorldAccessRight, hasWriteApplicationAccessRight, hasWriteOrganisationAccessRight, hasWriteWorldAccessRight, setPermissions` |

| Methods inherited from class java.lang.Object |
| --- |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Constructor Detail

### ExtendedFileAccessPermissions

```
public ExtendedFileAccessPermissions(boolean readWorldAccessRight,
                                     boolean writeWorldAccessRight,
                                     boolean readOrganisationAccessRight,
                                     boolean writeOrganisationAccessRight,
                                     boolean readApplicationAccessRight,
                                     boolean writeApplicationAccessRight,
                                     int[] otherOrganisationsReadAccessRights,
int[] otherOrganisationsWriteAccessRights)
```

This constructor encodes application, application organization, and world file access permissions as a set of booleans, and other organizations file access permissions as arrays of granted organization identifiers.

**Parameters:**

`readWorldAccessRight` - read access for all applications

`writeWorldAccessRight` - write access for all applications

`readOrganisationAccessRight` - read access for applications with the same organization as the granting application.

`writeOrganisationAccessRight` - write access for applications with the same organization as the granting application.

`readApplicationAccessRight` - read access for the owner.

`writeApplicationAccessRight` - write access for the owner.

`otherOrganisationsReadAccessRights` - array of other organization identifiers with read access. Applications with an organization identifier matching one of these organization identifiers will be given read access.

`otherOrganisationsWriteAccessRights` - array of other organization identifiers with write access. Applications with an organization identifier matching one of these organization identifiers will be given write access.

# Method Detail

### setPermissions

```
public void setPermissions(boolean readWorldAccessRight,
                           boolean writeWorldAccessRight,
                           boolean readOrganisationAccessRight,
                           boolean writeOrganisationAccessRight,
                           boolean readApplicationAccessRight,
                           boolean writeApplicationAccessRight,
                           int[] otherOrganisationsReadAccessRights,
                           int[] otherOrganisationsWriteAcessRights)
```

This method allows modification of the permissions on this instance of the ExtendedFileAccessPermission class.

**Parameters:**

`readWorldAccessRight` - read access for all applications

`writeWorldAccessRight` - write access for all applications
`readOrganisationAccessRight` - read access for organization
`writeOrganisationAccessRight` - write access for organization
`readApplicationAccessRight` - read access for the owner
`writeApplicationAccessRight` - write access for the owner
`otherOrganisationsReadAccessRights` - array of other organization identifiers with read access. Applications with an organization identifier matching one of these organization identifiers will be given read access.
`otherOrganisationsWriteAcessRights` - array of other organization identifiers with write access. Applications with an organization identifier matching one of these organization identifiers will be given write access.

### getReadAccessorganisationIds

`public int[] **getReadAccessorganisationIds**()`

Gets the array of organization identifiers with read permission.

**Returns:**

Array of organization identifiers with read permission.

### getWriteAccessorganisationIds

`public int[] **getWriteAccessorganisationIds**()`

Gets the array of organization identifiers with write permission.

**Returns:**

Array of organization identifiers with write permission.

**org.ocap.storage**

## Interface LogicalStorageVolume

```
public interface LogicalStorageVolume
```

This interface represents a logical volume on a storage device. Each `StorageProxy` corresponding to a storage device may contain LogicalStorageVolumes. A logical volume is a construct for organizing files on a disk and corresponds to a directory subtree that is treated as a whole for some purposes.

# Method Summary

| | |
|---|---|
| `ExtendedFileAccessPermissions` | **`getFileAccessPermissions`**`()`<br>Gets the file access permissions of the logical volume. |
| `java.lang.String` | **`getPath`**`()`<br>Gets the absolute path of the volume. |
| `StorageProxy` | **`getStorageProxy`**`()`<br>Gets the `StorageProxy` the volume is a part of. |
| `void` | **`setFileAccessPermissions`**`(`<br>`ExtendedFileAccessPermissions fap)`<br>Sets the file access permissions of the volume. |

# Method Detail

### getPath

`java.lang.String` **`getPath`**`()`

Gets the absolute path of the volume. This path must be unique across all LogicalStorageVolume instances regardless of the StorageProxy they are contained within.

**Returns:**

Absolute directory path of the volume.

### getStorageProxy

`StorageProxy` **`getStorageProxy`**`()`

Gets the `StorageProxy` the volume is a part of.

**Returns:**

Containing storage proxy.

### getFileAccessPermissions

`ExtendedFileAccessPermissions` **`getFileAccessPermissions`**`()`

Gets the file access permissions of the logical volume.

**Returns:**

File access permissions of the volume.

#### setFileAccessPermissions

`void` **`setFileAccessPermissions`**`(ExtendedFileAccessPermissions fap)`

Sets the file access permissions of the volume.

**Parameters:**

`fap` - New file access permissions.

**Throws:**

`java.lang.SecurityException` - if the caller is not the owner of the volume or does not have MonitorAppPermission("storage").

**org.ocap.storage**
## Interface RemovableStorageOption

**All Superinterfaces:**
StorageOption

```
public interface RemovableStorageOption
extends StorageOption
```

This interface represents a storage device that can be inserted into or removed from a bay while power is applied, e.g., memory stick. When such a device is inserted into or removed from its bay a `StorageManagerEvent` SHALL be generated with an event type of STORAGE_PROXY_CHANGED.

# Method Summary

| | |
|---|---|
| void | **eject**()<br>Prepares the storage device to be physically ejected from its bay in an implementation specific fashion, if applicable to the hardware. |
| boolean | **isPresent**()<br>Returns a presence indication for the removable device. |

# Method Detail

### eject

```
void eject()
```

Prepares the storage device to be physically ejected from its bay in an implementation specific fashion, if applicable to the hardware. If eject is not applicable to the storage device hardware this method does nothing and returns successfully

### isPresent

```
boolean isPresent()
```

Returns a presence indication for the removable device.

**Returns:**

True if a removable storage device is present in the corresponding bay, otherwise returns false.

**org.ocap.storage**
## Class StorageManager

```
java.lang.Object
  └ org.ocap.storage.StorageManager
```

```
public abstract class StorageManager
extends java.lang.Object
```

This class represents the storage manager which keeps track of the storage devices attached to the system.

| Method Summary | |
|---|---|
| abstract void | **addAvailableStorageListener**(AvailableStorageListener listener, int highWaterMark)<br>Adds a listener for high water mark reached in available persistent storage indicated by the dvb.persistent.root property. |
| abstract void | **addStorageManagerListener**(StorageManagerListener listener)<br>Adds a listener to receive StorageManagerEvents when a storage proxy is added, removed or changes state. |
| abstract long | **getAvailablePersistentStorage**()<br>Gets the available amount of persistent storage under the location indicated by the dvb.persistent.root property that is available to all OCAP-J applications. |
| static StorageManager | **getInstance**()<br>Gets the singleton instance of the storage manager. |
| abstract StorageProxy[] | **getStorageProxies**()<br>Gets the set of StorageProxy instances representing all of the currently attached or embedded storage devices. |
| abstract long | **getTotalPersistentStorage**()<br>Gets the total amount of persistent storage under the location indicated by the dvb.persistent.root property and that is usable by all OCAP-J applications. |
| abstract void | **removeAvailableStorageListener**(AvailableStorageListener listener)<br>Removes an available storage listener that was registered using the addAvailableStorageListener method. |
| abstract void | **removeStorageManagerListener**(StorageManagerListener listener)<br>Removes a listener so that it no longer receives StorageManagerEvents when storage proxies change. |

| Methods inherited from class java.lang.Object |
|---|
| equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Method Detail

### getInstance

```
public static StorageManager getInstance()
```

Gets the singleton instance of the storage manager. The singleton MAY be implemented using application or implementation scope.

**Returns:**

The storage manager.

### getStorageProxies

```
public abstract StorageProxy[] getStorageProxies()
```

Gets the set of `StorageProxy` instances representing all of the currently attached or embedded storage devices.

**Returns:**

An array of StorageProxy objects. If no application accessible storage proxies are available, returns a 0 length array.

### addStorageManagerListener

```
public abstract void
addStorageManagerListener(StorageManagerListener listener)
                                          throws
java.lang.IllegalArgumentException
```

Adds a listener to receive StorageManagerEvents when a storage proxy is added, removed or changes state.

**Parameters:**

`listener` - The storage manager listener to be added.

**Throws:**

`java.lang.IllegalArgumentException` - if the listener parameter has not been added.

### removeStorageManagerListener

```
public abstract void
removeStorageManagerListener(StorageManagerListener listener)
                                             throws
java.lang.IllegalArgumentException
```

Removes a listener so that it no longer receives StorageManagerEvents when storage proxies change.

**Parameters:**

`listener` - The storage manager listener to be removed.

**Throws:**

`java.lang.IllegalArgumentException` - if the listener parameter has not been added.

### getTotalPersistentStorage

```
public abstract long getTotalPersistentStorage()
```

Gets the total amount of persistent storage under the location indicated by the dvb.persistent.root property and that is usable by all OCAP-J applications. This value SHALL remain constant.

**Returns:**

Amount of total persistent storage in bytes.

#### getAvailablePersistentStorage

```
public abstract long getAvailablePersistentStorage()
```

Gets the available amount of persistent storage under the location indicated by the dvb.persistent.root property that is available to all OCAP-J applications. The value returned by this method can be incorrect as soon as this method returns and SHOULD be interpreted by applications as an approximation.

**Returns:**

Amount of available persistent storage in bytes.

#### addAvailableStorageListener

```
public abstract void
addAvailableStorageListener(AvailableStorageListener listener,
                                                int highWaterMark)
```

Adds a listener for high water mark reached in available persistent storage indicated by the dvb.persistent.root property. This is a system wide indication. Listeners are informed when a percentage of the total persistent storage has been allocated for application use. Listeners are only informed when the high water mark is reached or exceeded.

**Parameters:**

`listener` - The listener to add.

`highWaterMark` - Percentage of the available persistent storage remaining when the listener is to be informed. For instance, if the total available persistent storage is 1MB and the high water mark is 75 then high water listeners will be informed when 750KB have been allocated for application use.

**Throws:**

`java.lang.IllegalArgumentException` - if the listener parameter could not be added or is null.

#### removeAvailableStorageListener

```
public abstract void
removeAvailableStorageListener(AvailableStorageListener listener)
```

Removes an available storage listener that was registered using the `addAvailableStorageListener` method. If the parameter is not currently registered this method does nothing successfully.

**Parameters:**

`listener` - The listener to remove.

**Throws:**

`java.lang.IllegalArgumentException` - if the parameter is null.

**org.ocap.storage**

## Class StorageManagerEvent

```
java.lang.Object
  └ java.util.EventObject
      └ org.ocap.storage.StorageManagerEvent
```

**All Implemented Interfaces:**

java.io.Serializable

```
public class StorageManagerEvent
extends java.util.EventObject
```

Event sent to a `StorageManagerListener` registered with the `StorageManager` that a `StorageProxy` was added, removed or changed state.

## Field Summary

| | |
|---|---|
| `static int` | **`STORAGE_PROXY_ADDED`**<br>A StorageProxy was added. |
| `static int` | **`STORAGE_PROXY_CHANGED`**<br>A StorageProxy changed state. |
| `static int` | **`STORAGE_PROXY_REMOVED`**<br>A StorageProxy was removed. |

| **Fields inherited from class java.util.EventObject** |
|---|
| `source` |

## Constructor Summary

| |
|---|
| **`StorageManagerEvent`**`(StorageProxy proxy, int eventType)`<br>Constructs the event. |

## Method Summary

| | |
|---|---|
| `int` | **`getEventType`**`()`<br>Returns the event type. |
| `StorageProxy` | **`getStorageProxy`**`()`<br>Returns the `StorageProxy` that caused the event. |

| **Methods inherited from class java.util.EventObject** |
|---|
| `getSource, toString` |

| Methods inherited from class java.lang.Object |
| --- |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

# Field Detail

### STORAGE_PROXY_ADDED

```
public static final int STORAGE_PROXY_ADDED
```

A StorageProxy was added.

### STORAGE_PROXY_REMOVED

```
public static final int STORAGE_PROXY_REMOVED
```

A StorageProxy was removed.

### STORAGE_PROXY_CHANGED

```
public static final int STORAGE_PROXY_CHANGED
```

A StorageProxy changed state.

# Constructor Detail

### StorageManagerEvent

```
public StorageManagerEvent(StorageProxy proxy,
                           int eventType)
```

Constructs the event.

**Parameters:**

`proxy` - The `StorageProxy` that caused the event.

`eventType` - The type of event.

# Method Detail

### getEventType

```
public int getEventType()
```

Returns the event type. Possible values include STORAGE_PROXY_ADDED, STORAGE_PROXY_REMOVED or STORAGE_PROXY_CHANGED.

**Returns:**

The event type.

### getStorageProxy

```
public StorageProxy getStorageProxy()
```

Returns the `StorageProxy` that caused the event.

**Returns:**

The `StorageProxy` that caused the event.

**org.ocap.storage**

## Interface StorageManagerListener

**All Superinterfaces:**
java.util.EventListener

```
public interface StorageManagerListener
extends java.util.EventListener
```

This interface represents a listener for changes to the set of StorageProxies (`StorageProxy`). Each listener is notified when a storage proxy is added, removed or changes state.

# Method Summary

| | |
|---|---|
| void | **notifyChange**(StorageManagerEvent sme)<br>The implementation calls this method to inform the listener when a storage proxy was added, removed or changes state. |

# Method Detail

### notifyChange

```
void notifyChange(StorageManagerEvent sme)
```

The implementation calls this method to inform the listener when a storage proxy was added, removed or changes state. changed.

**Parameters:**
`sme` - Event indicating the change.

**See Also:**
`StorageManagerEvent`

**org.ocap.storage**

## Interface StorageOption

**All Known Subinterfaces:**

DetachableStorageOption, RemovableStorageOption

```
public interface StorageOption
```

This interface represents an option that is specific to a class of storage devices, e.g., detachable or DVR media-capable. It provides a mechanism for exposing special storage device characteristics and capabilities to applications.

**org.ocap.storage**
## Interface StorageProxy

**All Superinterfaces:**
org.davic.resources.ResourceProxy

```
public interface StorageProxy
extends org.davic.resources.ResourceProxy
```

This interface represents a persistent storage device. The current set of storage proxies is queried from the `StorageManager`.

A StorageProxy may contain one or more logical volumes (see org.ocap.storage.LogicalStorageVolume). A LogicalStorageVolume is a construct for organizing files on a disk and corresponds to a directory subtree that is treated as a whole for some purposes. A StorageProxy only represents the application visible storage on the device, i.e. it does not include portions of the device reserved for internal system use.

If this proxy represents a detachable or hot-pluggable device, the proxy is not listed until the device is connected. When a storage device is attached, the proxy is added to the list returned by the StorageManager and an appropriate event is sent to any `StorageManagerListener` registered with the `StorageManager`. When a storage device is no longer available for use (or for reactivation after being made detachable), the corresponding proxy is removed from the StorageManager's list and a `StorageManagerEvent` is sent to any StorageManagerListeners registered with the StorageManager.

StorageProxy extends ResourceProxy as an implementation convenience for resource contention handling. This ResourceProxy is not meant for access by applications and the StorageProxy.getClient method SHALL always return null.

# Field Summary

| | |
|---|---|
| `static byte` | **`BUSY`**<br>Returned by `getStatus()` to indicate that the device is busy, e.g., being initialized, configured, checked for consistency or being made ready to detach. |
| `static byte` | **`DEVICE_ERROR`**<br>Returned by `getStatus()` to indicate that the device is in an unrecoverable error state and cannot be used. |
| `static byte` | **`NOT_PRESENT`**<br>Returned by `getStatus()` to indicate that a detected storage device bay does not contain a removable storage device, i.e. |
| `static byte` | **`OFFLINE`**<br>Returned by `getStatus()` to indicate that the device is present but some other action is required before the device can be used (e.g., `DetachableStorageOption.makeReady()`). |
| `static byte` | **`READY`**<br>Returned by `getStatus()` to indicate that the device is initialized, mounted and ready for use. |
| `static byte` | **`UNINITIALIZED`**<br>Returned by `getStatus()` to indicate that the device is completely uninitialized and contains no existing data. |

## Field Summary

| | |
|---|---|
| static byte | **UNSUPPORTED_DEVICE**<br>Returned by `getStatus()` to indicate that the device that has been plugged in is not supported by the platform. |
| static byte | **UNSUPPORTED_FORMAT**<br>Returned by `getStatus()` to indicate that although the device is a supported type and model, it currently has a format, e.g., partitions or filesystems, that is not usable by the platform without reinitialization and the loss of the existing contents. |

## Method Summary

| | |
|---|---|
| LogicalStorageVolume | **allocateGeneralPurposeVolume**(java.lang.String name, ExtendedFileAccessPermissions fap)<br>Allocates a general purpose LogicalStorageVolume. |
| void | **deleteVolume**(LogicalStorageVolume vsp)<br>Deletes a LogicalStorageVolume. |
| java.lang.String | **getDisplayName**()<br>Gets a storage device name that can be displayed to a user for selection. |
| long | **getFreeSpace**()<br>Gets the available storage capacity in bytes. |
| java.lang.String | **getName**()<br>Gets the storage device name assigned by the implementation. |
| StorageOption[] | **getOptions**()<br>Gets the array of storage device options (e.g., DetachableStorageOption). |
| byte | **getStatus**()<br>Returns the status of the storage device. |
| boolean[] | **getSupportedAccessRights**()<br>Gets the permissions supported by this storage device. |
| long | **getTotalSpace**()<br>Gets the total storage capacity of the device in bytes. |
| LogicalStorageVolume[] | **getVolumes**()<br>Gets the set of logical volumes see present on the StorageProxy. |
| void | **initialize**(boolean userAuthorized)<br>Initializes the StorageProxy for use. |

| **Methods inherited from interface org.davic.resources.ResourceProxy** |
|---|
| getClient |

# Field Detail

### READY

`static final byte` **`READY`**

Returned by `getStatus()` to indicate that the device is initialized, mounted and ready for use.

### OFFLINE

`static final byte` **`OFFLINE`**

Returned by `getStatus()` to indicate that the device is present but some other action is required before the device can be used (e.g., `DetachableStorageOption.makeReady()`).

### BUSY

`static final byte` **`BUSY`**

Returned by `getStatus()` to indicate that the device is busy, e.g., being initialized, configured, checked for consistency or being made ready to detach. This value is not used to indicate that the device is currently reading or writing data.

### UNSUPPORTED_DEVICE

`static final byte` **`UNSUPPORTED_DEVICE`**

Returned by `getStatus()` to indicate that the device that has been plugged in is not supported by the platform.

### UNSUPPORTED_FORMAT

`static final byte` **`UNSUPPORTED_FORMAT`**

Returned by `getStatus()` to indicate that although the device is a supported type and model, it currently has a format, e.g., partitions or filesystems, that is not usable by the platform without reinitialization and the loss of the existing contents.

### UNINITIALIZED

`static final byte` **`UNINITIALIZED`**

Returned by `getStatus()` to indicate that the device is completely uninitialized and contains no existing data. It must be initialized by calling the initialize method to make the device is usable.

### DEVICE_ERROR

`static final byte` **`DEVICE_ERROR`**

Returned by `getStatus()` to indicate that the device is in an unrecoverable error state and cannot be used.

### NOT_PRESENT

`static final byte` **`NOT_PRESENT`**

Returned by `getStatus()` to indicate that a detected storage device bay does not contain a removable storage device, i.e. `StorageProxy` containing a `RemovableStorageOption`.

# Method Detail

### getName

`java.lang.String` **`getName`**`()`

Gets the storage device name assigned by the implementation. This name must be unique across all storage devices. The name can be used to determine equality between two storage devices, but does not contain path information.

**Returns:**

The name of the resource represented by the proxy.

### getDisplayName

`java.lang.String` **`getDisplayName`**`()`

Gets a storage device name that can be displayed to a user for selection. The implementation must keep this name at or below 40 characters in length. This name should match naming conventions displayed to the consumer via any implementation specific setup and configuration menus.

**Returns:**

The display name of the resource represented by the proxy.

### getOptions

`StorageOption[]` **`getOptions`**`()`

Gets the array of storage device options (e.g., `DetachableStorageOption`).

**Returns:**

The array of StorageOptions associated with this StorageProxy.

### getStatus

`byte` **`getStatus`**`()`

Returns the status of the storage device. An application can be notified of changes in the status of storage proxies by registering a `StorageManagerListener` with `StorageManager.addStorageManagerListener()` .

### getTotalSpace

`long` **`getTotalSpace`**`()`

Gets the total storage capacity of the device in bytes. Storage that is reserved for system use is not included in this number.

**Returns:**

Total storage capacity in bytes.

### getFreeSpace

`long` **`getFreeSpace`**`()`

Gets the available storage capacity in bytes. The value returned may already have changed by the time this method returns because other applications or the system may be writing files, deleting files, or otherwise allocating space.

**Returns:**

Available storage capacity in bytes.

#### getVolumes

`LogicalStorageVolume[] getVolumes()`

Gets the set of logical volumes see present on the StorageProxy. If a StorageProxy has no logical volumes present, one or more must be created before the device may be used for application storage.

**Returns:**

The partitioned storage volumes.

#### getSupportedAccessRights

`boolean[] getSupportedAccessRights()`

Gets the permissions supported by this storage device.

**Returns:**

An array of booleans indicating which access rights are supported where location 0 is world read access right, 1 is world write access right, 2 is application read access right, 3 is application write access right, 4 is application's organization read access right, 5 is application's organization write access right, 6 is other organization read access right, and 7 is other organization write access right. If the boolean for one of the access rights is true, the storage device supports it, otherwise the storage device does not support that access right.

#### allocateGeneralPurposeVolume

`LogicalStorageVolume allocateGeneralPurposeVolume(java.lang.String name,`

`ExtendedFileAccessPermissions fap)`

Allocates a general purpose `LogicalStorageVolume`. A general purpose volume can be accessed through file locators and java.io with the absolute path retrieved from `LogicalStorageVolume.getPath()`. Specialized storage proxies may support other types of volumes, such as media volumes used to store DVR content. The volume is owned by the application that allocated it (see `deleteVolume(org.ocap.storage.LogicalStorageVolume)`).

**Parameters:**

`name` - Name of the new LogicalStorageVolume. Must be unique on this StorageProxy.

`fap` - Application access permissions of the new LogicalStorageVolume. Applies to the last directory in the path returned by getPath, which is equivalent to the name parameter.

**Returns:**

Allocated volume storage proxy.

**Throws:**

`java.lang.IllegalArgumentException` - if the name does not meet the from specified by chapter 16 section regarding Files and File Names, or if the name is not unique, or if the storage device does not support an access permission specified in the fap parameter.

`java.io.IOException` - if the storage device represented by the StorageProxy is read-only based on a hardware constraint.

`java.lang.SecurityException` - if the calling application does not have persistent storage permission as requested by its permission request file.

#### deleteVolume

`void` **`deleteVolume`**`(LogicalStorageVolume vsp)`

Deletes a `LogicalStorageVolume`. Only the owning application or a privileged application with MonitorAppPermission("storage") may delete a volume. This causes all of the file and directories within the volume to be destroyed.

**Parameters:**

`vsp` - LogicalStorageVolume to delete.

**Throws:**

`java.lang.SecurityException` - if the calling application is not the owner of the volume or an application with MonitorAppPermission("storage").

#### initialize

`void` **`initialize`**`(boolean userAuthorized)`

Initializes the StorageProxy for use. This method is usually invoked on the proxy for a newly attached storage device which is not currently suitable for use, e.g., is in the `UNSUPPORTED_FORMAT` state. It is only required to be effective on detachable storage devices, but may be implemented for other types of devices as well. Successful invocation of this method destroys all application visible contents of the device and should not be called unless the application has determined, e.g., by prompting the user, that it is safe to do so. If the StorageProxy was in the `READY` state and has storage visible to the application, access to that storage is removed and the StorageProxy enters the `BUSY` state until this method returns.

**Parameters:**

`userAuthorized` - True if the application has received authorization from the user for the destruction of the contents of this device. The implementation may use this to determine whether it needs to perform additional user prompting.

**Throws:**

`java.lang.SecurityException` - if the calling application does not have MonitorAppPermission("storage").

`java.lang.IllegalStateException` - if the system is unable to initialize the storage device. If the device was in the `UNINITIALIZED` state and the error is permanent, the StorageProxy status is set to `DEVICE_ERROR`.

# Annex W OCAP 1.1 Test API

## Package org.ocap.test

OCAP testing communications.

| Class Summary | |
|---|---|
| **OCAPTest** | The purpose of this class is to provide a very simple communication channel between the OCAP implementation under test (IUT) and the test server. |

## Package org.ocap.test Description

OCAP testing communications.

OCAP testing requires that test applications executing on OCAP devices be able to communicate with an Automated Test Environment (ATE) in a manufacturer-independent fashion. This differs from the MHP test environment, in which MHP device manufacturers provide manufacturer-specific implementations to communicate between an ATE and an MHP device. The need to communicate with an ATE in a manufacturer- independent manner means that OCAP devices must communicate with the ATE using a common, defined communications mechanism.

Since OCAP devices by definition must support TCP/IP connectivity, TCP/IP sockets have been chosen as the mechanism by which OCAP devices will communicate with an OCAP ATE. The communications channel discussed here SHALL be that which provides TCP/IP communications via the RF connector labeled 'Cable In' on the OCAP Host Device.

## OCAP compliance testing

Due to the large number of OCAP Host Devices which are expected to be made available by manufacturers during certification testing "waves", OCAP compliance testing will need to be able to take advantage of a common, scalable hardware infrastructure.
The proposed connectivity diagram is shown below in Figure 1:

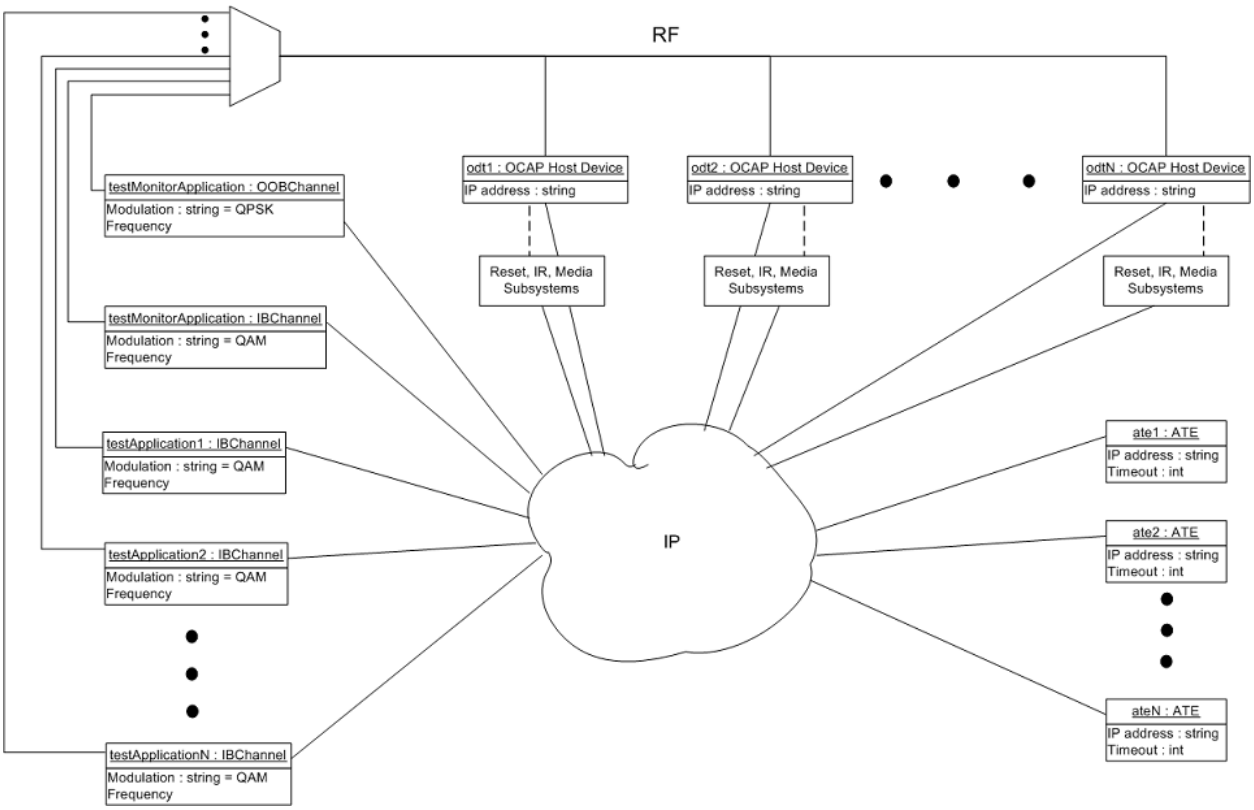



***Figure W–1 - Connectivity Diagram***

This diagram is intended to show how multiple ATEs can be connected on a common IP network with multiple OCAP Host Devices. On the left is a single Out-Of-Band (OOB) channel which can stream a common test monitor application to all OCAP Host Devices being tested. In-band (IB) delivery channels are used to stream test applications, and are controlled by a particular ATE instance. All of these IB and OOB channels are muxed onto a common RF cable which is shared by all OCAP Host Devices being tested.
Each OCAP Host Device that is being tested is associated with a test harness comprised of reset, IR, and media capture subsystems. These subsystems are controlled by a particular ATE instance. The ATE instances are shown on the right side of the drawing.
At configuration time, an instance of the ATE is associated with a particular IB delivery channel, an OCAP Host Device being tested, and its associated test harness components.
During compliance testing, a particular OCAP Host Device establishes a communication channel with a particular instance of the ATE. The ATE is responsible for preparing test applications to be loaded onto the OCAP Host Device (via its associated IB delivery channel), and for responding to requests made by test applications executing on the OCAP Host Device being tested.

The scenario for establishing communications between an OCAP Host Device being tested and an ATE is as follows (details are addressed in the CableLabs' ATE documentation):
1. The ATE uses the "reset for next test" mechanism to set the OCAP Host Device being tested into a known default state, ready to receive the test-application. This involves a power-cycle of the OCAP Host Device being tested.
2. The ATE begins sending IP datagrams to the IP address of the OCAP Host Device it is associated with.
3. The OCAP Host Device powers up out of reset and reads an XAIT from the out-of-band communications channel, signaling the OCAP Host Device to load a Test Monitor Application (an unbound application) from an in-band communications channel.
4. The Test Monitor Application invokes `OcapSystem.monitorConfiguredSignal()` to enable conformance testing APIs (`org.ocap.test.OCAPTest`). When monitorConfiguredSignal is called, the implementation will enable the conformance testing APIs (see org.ocap.OcapSystem.monitorConfiguredSignal for details).
6. Test Monitor Application requests configuration parameters including information on the initial channel to tune

to using the communications channel established in step 5c.
7. Tuning to the specified channel initiates a service containing the bound testlet application.
8. Test application executes on OCAP Host Device, communicating with the ATE as required via the communications channel opened up in step 5.c.

## OCAP testing communications

OCAP testing requires that test applications executing on OCAP devices be able to communicate with an Automated Test Environment (ATE) in a manufacturer-independent fashion. The need to communicate with an ATE in a manufacturer-independent manner means that OCAP Host Devices must communicate with the ATE using a common, bi-directional communications channel. In an effort to isolate OCAP Host Device implementations from the need to implement communications protocols which are necessary only for use in compliance testing, the class `org.ocap.test.OCAPTest` has been defined. This class presents an API to test framework classes which can be used for compliance testing. The conceptual diagram of this communications structure is presented below:

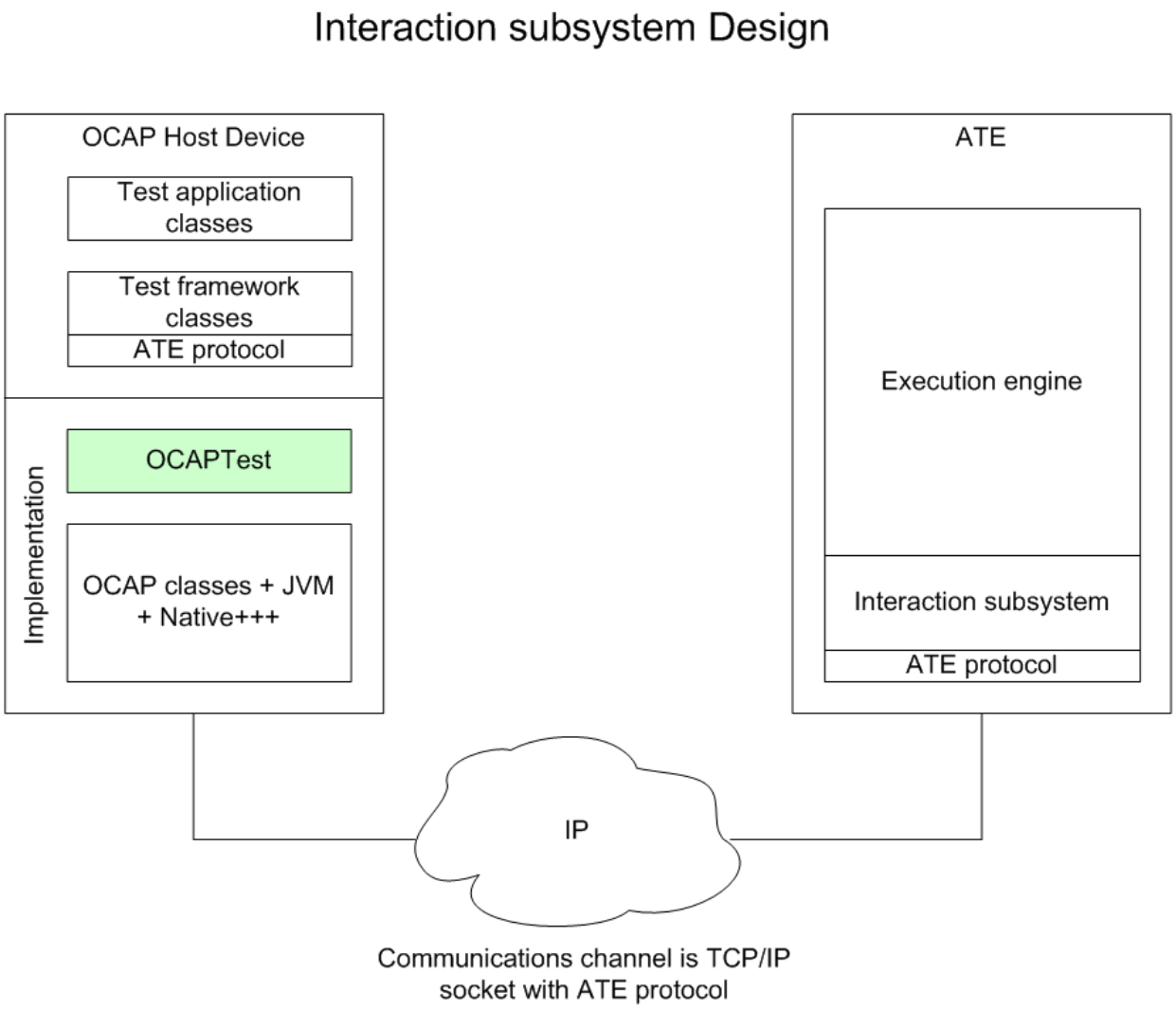



***Figure W–2 - Interaction Subsystem Design Diagram***

**org.ocap.test**
### Class OCAPTest

```
java.lang.Object
  └ org.ocap.test.OCAPTest

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

The purpose of this class is to provide a very simple communication channel between the OCAP implementation under test (IUT) and the test server. The functionality of this class is intentionally limited to sending/receiving raw messages via TCP/IP or UDP/IP protocol. This approach decouples the lower communications channel from the higher messaging protocol used to exchange messages between the test client and the test server. As a result, the implementation will be isolated from any future messaging protocol changes which may be required for future testing needs.
This class does not provide a means for concurrently executing test applications to negotiate reservation of the interaction channel used to exchange message with the test environment host. Test application authors should ensure that concurrently executing test applications correctly interoperate in their use of this channel.
The implementation of this class MUST use TCP/IP or UDP/IP protocol to establish a socket connection to the test server via the RF connector labeled 'Cable In' on the OCAP Host Device. See the `getProtocol()` method.

## Field Summary

| | |
|---|---|
| static int | **MAX_MESSAGE_LENGTH**<br>Maximum length of messages exchanged between ATE and a test application. |
| static byte | **MESSAGE_TERMINATION_BYTE**<br>Message termination byte. |
| static int | **TCP**<br>Indicates that an OCAPTest instance is configured to use TCP/IP protocol. |
| static int | **UDP**<br>Indicates that an OCAPTest instance is configured to use UDP/IP protocol. |

## Constructor Summary

| |
|---|
| **OCAPTest**() |

## Method Summary

| | |
|---|---|
| static int | **getProtocol**()<br>Returns a current protocol that is used by an OCAPTest instance. |
| static byte[] | **receive**()<br>Receive a byte array from ATE via TCP/IP. |
| static byte[] | **receiveUDP**()<br>Receive a message from ATE via UDP/IP. |
| static void | **send**(byte[] rawMessage)<br>Send a specified byte array to ATE via TCP/IP protocol. |
| static void | **sendUDP**(byte[] rawMessage)<br>Send rawMessage to ATE via UDP/IP. |

| Methods inherited from class java.lang.Object |
| --- |
| `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait` |

# Field Detail

### MESSAGE_TERMINATION_BYTE

`public static final byte` **`MESSAGE_TERMINATION_BYTE`**

Message termination byte. Used to identify the end of a sequence of message bytes between ATE and a test application. For the `send(byte[])` and `receive()` methods the specified rawMessage and the returned bytes shall not contain MESSAGE_TERMINATION_BYTE except indicating message termination. For the `sendUDP(byte[])` and `receiveUDP()` methods the specified rawMessage and the returned bytes may contain MESSAGE_TERMINATION_BYTE to process acknowledge protocol.

### MAX_MESSAGE_LENGTH

`public static final int` **`MAX_MESSAGE_LENGTH`**

Maximum length of messages exchanged between ATE and a test application.

### UDP

`public static final int` **`UDP`**

Indicates that an OCAPTest instance is configured to use UDP/IP protocol.

### TCP

`public static final int` **`TCP`**

Indicates that an OCAPTest instance is configured to use TCP/IP protocol.

# Constructor Detail

### OCAPTest

`public` **`OCAPTest`**`()`

# Method Detail

### send

```
public static void send(byte[] rawMessage)
                throws java.io.IOException
```

Send a specified byte array to ATE via TCP/IP protocol. The message MUST NOT be altered before or while sending.

A test application specifies a byte array that is less than MAX_MESSAGE_LENGTH to the rawMessage

parameter. The byte array shall not contain MESSAGE_TERMINATION_BYTE. This method SHALL NOT make any assumptions as to the format of the content of the rawMessage other than it can't contain a byte with the value MESSAGE_TERMINATION_BYTE except message termination. After the specified rawMessage are sent, `MESSAGE_TERMINATION_BYTE` will be sent by this method, indicating to the ATE that the rawMessage is complete, and ready for parsing.
In case of buffered connections, the buffer MUST be flushed upon exiting this method.
**Parameters:**
`rawMessage` - a byte array of the raw message to be sent to ATE via TCP/IP protocol.
**Throws:**
`java.lang.IllegalArgumentException` - If rawMessage contains a byte with the value MESSAGE_TERMINATION_BYTE.
`java.io.IOException` - If there is any problem with I/O operations or an interaction channel has not been initialized.

#### receive

```
public static byte[] receive()
                     throws java.io.IOException
```

Receive a byte array from ATE via TCP/IP.
The message MUST NOT be altered during or after reception.
This is a blocking method which waits for an entire of original message from ATE. The implementation should accumulate bytes from the test server until the `MESSAGE_TERMINATION_BYTE` is received. The all received bytes MUST then be returned to the caller as a `byte[]`. The termination character is not to be included in the returned byte array. There should be no other assumptions as to the format or content of the message bytes.
The maximum number of returned byte array is `MAX_MESSAGE_LENGTH`. Any bytes received beyond MAX_MESSAGE_LENGTH should be discarded. In any event, this method must not return until a `MESSAGE_TERMINATION_BYTE` has been encountered.
**Returns:**
a byte array coming from ATE via TCP/IP protocol. The termination character is not included.
**Throws:**
`java.io.IOException` - If there is any problem with I/O operations or an interaction channel has not been initialized.

#### getProtocol

```
public static int getProtocol()
```

Returns a current protocol that is used by an OCAPTest instance. The current protocol shall matches with a protocol specified by a ppp value of an "ate:a.b.c.d:xxxx:ppp" message from ATE.
**Returns:**
a current protocol that is used by an OCAPTest instance. Either `UDP` or `TCP` constant.

#### receiveUDP

```
public static byte[] receiveUDP()
                        throws java.io.IOException
```

Receive a message from ATE via UDP/IP. The ATE's IP address is specified by an "ate:a.b.c.d:xxxx:ppp" message from ATE. This method simply returns a payload bytes in a UDP packet without modification, i.e., the OCAPTest doesn't concatenate a sequence of messages. It is responsibility of a caller of this method to concatenate received byte arrays into an original message according to ATE acknowledge protocol. This method blocks the thread of a caller until receiving a UDP packet.
**Returns:**

byte a payload bytes in a UDP packet. The byte length must be less than a max length limited by the interaction channel and it is responsibility of a caller of this method to do so.

**Throws:**

`java.io.IOException` - If there is any problem with I/O operations or an interaction channel has not been initialized.

### sendUDP

```
public static void sendUDP(byte[] rawMessage)
                    throws java.io.IOException
```

Send rawMessage to ATE via UDP/IP. The ATE's IP address is specified by an "ate:a.b.c.d:xxxx:ppp" message from ATE. This method shall not divide the specified rawMessage into some UDP packets. The specified rawMessage shall be sent in a single UDP packet.

**Parameters:**

`rawMessage` - byte data to be sent. The byte length must be less than a max length limited by the interaction channel and it is responsibility of a caller of this method to do so.

**Throws:**

`java.io.IOException` - If there is any problem with I/O operations or an interaction channel has not been initialized.

# Annex X OCAP 1.1 Digital Program Insertion API

## Package org.ocap.dpi

| Interface Summary | |
|---|---|
| **SwitchEngineListener** | This class represents a listener for Switch Engine events. |

| Class Summary | |
|---|---|
| **SwitchEngineManager** | An application may use this class to control the behavior of the switch engine. |
| **SwitchInstruction** | A SwitchInstruction encapsulates the information needed by the switch engine to perform a switch operation. |

| Exception Summary | |
|---|---|
| **SwitchEngineException** | Derived from Exception and allows an exception to pass back the SwitchInstruction associated with the Exception. |

**org.ocap.dpi**

## Class SwitchEngineException

```
java.lang.Object
  └ java.lang.Throwable
      └ java.lang.Exception
          └ org.ocap.dpi.SwitchEngineException
```

**All Implemented Interfaces:**

java.io.Serializable

```
public class SwitchEngineException
extends java.lang.Exception
```

Derived from Exception and allows an exception to pass back the SwitchInstruction associated with the Exception.

| Constructor Summary |
|---|
| **SwitchEngineException**() |

# Method Summary

| | |
|---|---|
| SwitchInstruction[] | **SwitchInstructionException**()<br>Returns the SwitchInstruction(s) that caused the exception. |

| Methods inherited from class java.lang.Throwable |
|---|
| fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString |

| Methods inherited from class java.lang.Object |
|---|
| equals, getClass, hashCode, notify, notifyAll, wait, wait, wait |

# Constructor Detail

### SwitchEngineException

public **SwitchEngineException**()

# Method Detail

### SwitchInstructionException

public SwitchInstruction[] **SwitchInstructionException**()

Returns the SwitchInstruction(s) that caused the exception.

**Returns:**

SwitchInstruction[] SwitchInstructiion(s) that caused the Exception to be thrown.

**org.ocap.dpi**

## Interface SwitchEngineListener

**All Superinterfaces:**
java.util.EventListener

```
public interface SwitchEngineListener
extends java.util.EventListener
```

This class represents a listener for Switch Engine events.

# Field Summary

| | |
|---|---|
| static int | **COMPONENT_START**<br>For an L1 Switch a component switch has been completed. |
| static int | **INSERTION_START**<br>Switch Instruction was successfully executed and an insertion was accomplished. |
| static int | **INSTRUCTION_EXPIRED**<br>Switch Instruction was removed because it has passed its valid time based on it's ExpirationTime. |
| static int | **MISSED_TRIGGER**<br>The Trigger is position_in_set = 2 or greater and the Switch Instruction's trigger with position_in_set = 1 has not be processed for this Switch set. |
| static int | **NO_TARGET**<br>No target for insertion was found at the Locator included in the SwitchInstruction. |
| static int | **UNARMED_TRIGGER**<br>A Trigger was detected but but no matching SwitchInstruction was found. |

# Method Summary

| | |
|---|---|
| void | **notifySwitchEngineEvent**(SwitchInstruction SwitchInstruction, int Reason)<br>Method called by the Switch Engine implementation when a Switch Engine event occurs, if the Targeting Engine application has registered as a listener for the event. |

# Field Detail

### UNARMED_TRIGGER

```
static final int UNARMED_TRIGGER
```

A Trigger was detected but but no matching SwitchInstruction was found.

#### MISSED_TRIGGER

`static final int` **`MISSED_TRIGGER`**

The Trigger is position_in_set = 2 or greater and the Switch Instruction's trigger with position_in_set = 1 has not be processed for this Switch set.

#### NO_TARGET

`static final int` **`NO_TARGET`**

No target for insertion was found at the Locator included in the SwitchInstruction.

#### INSTRUCTION_EXPIRED

`static final int` **`INSTRUCTION_EXPIRED`**

Switch Instruction was removed because it has passed its valid time based on it's ExpirationTime. This may not be an error if it is for a different source than has been tuned during the Insertion Group.

#### INSERTION_START

`static final int` **`INSERTION_START`**

Switch Instruction was successfully executed and an insertion was accomplished.

#### COMPONENT_START

`static final int` **`COMPONENT_START`**

For an L1 Switch a component switch has been completed.

# Method Detail

#### notifySwitchEngineEvent

```
void notifySwitchEngineEvent(SwitchInstruction SwitchInstruction,
                             int Reason)
```

Method called by the Switch Engine implementation when a Switch Engine event occurs, if the Targeting Engine application has registered as a listener for the event.

**Parameters:**

`SwitchInstruction` - The SwitchInstruction that is associated with this Switch Engine event.

`Reason` - The reason for the event notification.

- UNARMED_TRIGGER - A Trigger was detected but no matching SwitchInstruction was found.
- MISSED_TRIGGER - The Trigger is position_in_set = 2 or greater and the trigger where position_in_set = 1, was not be processed for this Switch Group.
- NO_TARGET - No target for insertion was found at the Locator(s) included in the SwitchInstruction.
- INSTRUCTION_EXPIRED - Switch Instruction was removed because it has passed its valid time based on its ExpirationTime.
- INSERTION_START - Switch has been accomplished.

- COMPONENT_START - For an L1 Switch a component switch has been completed.

**org.ocap.dpi**

## Class SwitchEngineManager

```
java.lang.Object
  └ org.ocap.dpi.SwitchEngineManager
```

```
public abstract class SwitchEngineManager
extends java.lang.Object
```

An application may use this class to control the behavior of the switch engine.

# Method Summary

| | |
|---|---|
| abstract void | **addInstruction**(SwitchInstruction[] switchInstructions)<br>Add SwitchInstruction(s) to the Switch Engine actve list. |
| static SwitchEngineManager | **getInstance**()<br>gets the singleton instance of the Switch Engine manager for use by a priveleged application. |
| abstract SwitchInstruction[] | **getInstruction**(int[] SwitchIDs, int SourceID)<br>The Switch Engine SHALL return the Switch Instruction(s) matching the passed in SwitchID and SourceID. |
| abstract void | **removeInstruction**(SwitchInstruction[] switchInstructions)<br>The Switch Engine SHALL remove the passed in Switch Instruction(s) from the Switch Engine active list. |
| abstract void | **removeSwitchEngineListener**(SwitchEngineListener Listener)<br>Removes the previously added SwitchEngineListener. |
| abstract void | **setSwitchEngineListener**(SwitchEngineListener Listener, byte[] Filter)<br>Adds the listener for Switch Engine events. |

| Methods inherited from class java.lang.Object |
|---|
| equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Method Detail

### getInstance

```
public static SwitchEngineManager getInstance()
```

gets the singleton instance of the Switch Engine manager for use by a priveleged application.

**Throws:**

`java.lang.SecurityException` - if the calling application does not have MonitorAppPermission("dpi").

#### setSwitchEngineListener

```
public abstract void setSwitchEngineListener(SwitchEngineListener Listener,
                                             byte[] Filter)
```

Adds the listener for Switch Engine events.

**Parameters:**

`Listener` - Is an instance of a SwitchEngineListener whose SwitchEngineListener method SHALL be called when a Switch Engine event occurs.

`Filter` - A byte array of filters enabling any or all of the following Switch Engine events.

- UNARMED_TRIGGER
- MISSED_TRIGGER
- NO_TARGET
- INSTRUCTION_EXPIRED
- INSERTION_START
- COMPONENT_START

**Throws:**

`java.lang.IllegalArgumentException` - Listener is null

#### removeSwitchEngineListener

```
public abstract void removeSwitchEngineListener(SwitchEngineListener Listener)
```

Removes the previously added SwitchEngineListener.

**Parameters:**

`Listener` - is the Switch Engine Listener to disable. Does nothing if it was never added, has been removed, or is null.

#### addInstruction

```
public abstract void addInstruction(SwitchInstruction[] switchInstructions)
```

Add SwitchInstruction(s) to the Switch Engine actve list.

If TriggerSwitch is FALSE the Switch Engine SHALL perform an L0 Switch when time is at or past the SwitchStartTime in the SwitchInstruction. For all other SwitchInstructions the Switch Engine SHALL store the Switch Instruction(s) in its active list. For any Switch Instruction that have an identical SwitchID and SourceID to an already existing Switch Instruction the Switch Engine SHALL replace the Switch Instruction and a Duplicate Switch Instruction Exception is thrown.

**Parameters:**

`switchInstructions` -

**Throws:**

`java.lang.SecurityException` - Calling application does not have MonitorAppPermission("dpi")

`java.lang.IllegalArgumentException` - - If the BreakID and SourceID do not identify a SwitchInstruction.

`java.lang.IllegalArgumentException` - Negative SourceID

`java.lang.IllegalArgumentException` - ExpirationTime passed
`java.lang.IllegalArgumentException` - Unknown InsertionOptions
`java.lang.IllegalArgumentException` - SwitchEndTime is before SwitchStartTime
`java.lang.IllegalArgumentException` - Duplicate SwitchInstruction
`InvalidLocator` - if Locator is invalid
`InvalidLocator` - Locator is null
`InvalidLocator` - Locator not found

#### removeInstruction

```
public abstract void removeInstruction(SwitchInstruction[] switchInstructions)
```

The Switch Engine SHALL remove the passed in Switch Instruction(s) from the Switch Engine active list. If any argument is -1 then the Switch Engine SHALL treated it as a “wild card” such that the Switch Engine shall process all Switch Instructions match that field. (Ex. if SourceID = -1 then remove the Switch Instructions matching the SwitchID and all SourceID's).

**Parameters:**

`switchInstructions` - See SwitchInstruction class.

**Throws:**

`java.lang.IllegalArgumentException` - - If the BreakID and SourceID do not identify a SwitchInstruction.

`java.lang.SecurityException` - Calling application does not have MonitorAppPermission("dpi")

#### getInstruction

```
public abstract SwitchInstruction[] getInstruction(int[] SwitchIDs,
                                                   int SourceID)
```

The Switch Engine SHALL return the Switch Instruction(s) matching the passed in SwitchID and SourceID. If any argument is -1 then the Switch Engine SHALL treat it as a “wild card” such that all Switch Instructions match that field are processed. (Ex. if SourceID = -1 then get the Switch Instructions matching the SwitchID and all SourceID's).

**Parameters:**

`SwitchIDs` - One or more numbers uniquely identifying the SwitchInstructions to return.

`SourceID` - Source Identifier for the SwitchInstructions.

**Returns:**

SwitchInstruction An object that is created by the DPI Targeting Engine and passed to the implementation via the Switch engine API. The object SHALL be copied by the method and can't be changed after loaded.

**Throws:**

`java.lang.IllegalArgumentException` - - If the SwitchID and SourceID do not identify a SwitchInstruction.

`java.lang.SecurityException` - Calling application does not have MonitorAppPermission("dpi")

**org.ocap.dpi**

## Class SwitchInstruction

```
java.lang.Object
  └ org.ocap.dpi.SwitchInstruction
```

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

A SwitchInstruction encapsulates the information needed by the switch engine to perform a switch operation. A switch operation is performed according to the SwitchInstruction on receipt of a DPI trigger or when a point on a metadata timeline is reached.

Switches may be trigger based or time based. A SwitchInstruction MAY identify a set of triggers, and/or identify a segment of time during which a switch may occur.

A SwitchInstruction object MAY be instantiated by a targeting engine application and passed to the implementation via SwitchEngineManager.addInstruction().

For the attributes of the SwitchInstruction see the SwitchInstruction constructor parameter list.

## Field Summary

| | |
|---|---|
| static int | **GO_TO_BLACK**<br>Go to black at SwitchEndTime even if insert program is not complete. |
| static int | **PLAY_TO_COMPLETION**<br>Play inserted program to completion under all circumstances. |
| static int | **RETURN_TO_PROGRAM**<br>Return to Program at SwitchEndTime even if inserted content is not complete. |

## Constructor Summary

| |
|---|
| **SwitchInstruction**()<br>SwitchInstruction constructor. |
| **SwitchInstruction**(int SourceID, javax.tv.locator.Locator[] Locator, int SwitchID, byte GroupSize, byte PositionInGroup, int SwitchStartTime, int SwitchEndTime, int SwitchWindowEndTime, int ExpirationTime, boolean TriggerSwitch, int InsertionOptions, boolean InsertionIndivisible, boolean EndOfInsertion)<br>SwitchInstruction constructor. |

## Method Summary

| | |
|---|---|
| byte[] | **getEndOfInsertion**()<br>Retrieve the attribute EndOfInsertion. |
| int | **getExpirationTime**()<br>Retrieve the attribute ExpirationTime. |

| Method Summary | |
|---|---|
| byte | **getGroupSize**()<br>Retrieve the attribute GroupSize. |
| boolean | **getInsertionIndivisible**()<br>Retrieve the attribute InsertionIndivisible. |
| int | **getInsertionOptions**()<br>Retrieve the attribute InsertionOptions. |
| javax.tv.locator.Locator[] | **getLocator**()<br>Retrieve the attribute Locator. |
| byte | **getPositionInGroup**()<br>Retrieve the attribute PositionInGroup. |
| int | **getSourceID**()<br>Retrieve the attribute SourceID. |
| int | **getSwitchEndTime**()<br>Retrieve the attribute SwitchEndTime. |
| int | **getSwitchID**()<br>Retrieve the attribute SwitchID. |
| int | **getSwitchStartTime**()<br>Retrieve the attribute SwitchStartTime. |
| int | **getSwitchWindowEndTime**()<br>Retrieve the attribute SwitchWindowEndTime. |
| boolean | **getTriggerSwitch**()<br>Get attribute TriggerSwitch. |
| void | **setEndOfInsertion**(boolean EndOfInsertion)<br>Set attribute EndOfInsertion to specified parameter. |
| void | **setExpirationTime**(int ExpirationTime)<br>Set attribute ExpirationTime to specified parameter. |
| void | **setGroupSize**(byte GroupSize)<br>Set attribute GroupSize to specified parameter. |
| void | **setInsertionIndivisible**(boolean InsertionIndivisible)<br>Set attribute InsertionIndivisible to True or False. |
| void | **setInsertionOptions**(int InsertionOptions)<br>Set attribute InsertionOptions to specified parameter. |
| void | **setLocator**(javax.tv.locator.Locator[] Locator)<br>Set attribute Locator to specified parameter. |
| void | **setPositionInGroup**(byte PositionInGroup)<br>Set attribute PositionInGroup to specified parameter. |
| void | **setSourceID**(int SourceID)<br>Set attribute SourceID to specified parameter. |
| void | **setSwitchEndTime**(int SwitchEndTime)<br>Set attribute SwitchEndTime to specified parameter. |

# Method Summary

| | |
|---|---|
| void | **setSwitchID**(int SwitchID)<br>Set attribute SwitchID to specified parameter. |
| void | **setSwitchStartTime**(int SwitchStartTime)<br>Set attribute SwitchStartTime to specified parameter. |
| void | **setSwitchWindowEndTime**(int SwitchWindowEndTime)<br>Set attribute SwitchWindowEndTime to specified parameter. |
| void | **setTriggerSwitch**(boolean TriggerSwitch)<br>Set attribute TriggerSwitch. |

| **Methods inherited from class java.lang.Object** |
|---|
| equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Field Detail

### GO_TO_BLACK

```
public static final int GO_TO_BLACK
```
Go to black at SwitchEndTime even if insert program is not complete.

### RETURN_TO_PROGRAM

```
public static final int RETURN_TO_PROGRAM
```
Return to Program at SwitchEndTime even if inserted content is not complete.

### PLAY_TO_COMPLETION

```
public static final int PLAY_TO_COMPLETION
```
Play inserted program to completion under all circumstances.

# Constructor Detail

### SwitchInstruction

```
public SwitchInstruction()
```
SwitchInstruction constructor. Can be used to Instantiate a SwitchInstruction w/o initializing the attributes.

### SwitchInstruction

```
public SwitchInstruction(int SourceID,
                         javax.tv.locator.Locator[] Locator,
                         int SwitchID,
                         byte GroupSize,
```

```
                        byte PositionInGroup,
                        int SwitchStartTime,
                        int SwitchEndTime,
                        int SwitchWindowEndTime,
                        int ExpirationTime,
                        boolean TriggerSwitch,
                        int InsertionOptions,
                        boolean InsertionIndivisible,
                        boolean EndOfInsertion)
```

SwitchInstruction constructor. Can be used to initialize all attributes in a SwitchInstruction in place of the set methods.

**Parameters:**

`SourceID` - The source id of the service on which this instruction may cause a switch.

`Locator` - A Locator that identifies the component or program for insertion.

`SwitchID` - Switch Identifier. Matches the trigger identifier within a set of DPI triggers (synchronized_event_id).

`GroupSize` - Indicates the number of switches within a related group. A group corresponds to an advertising break, which may contain several ad spots.

`PositionInGroup` - Indicates the position of this switch within a group of insertions, zero based.

`SwitchStartTime` - Point on a metadata timeline at which an L0 switch may occur.

`SwitchEndTime` - Point on a metadata timeline at which the insertion segment is over and the switch engine should present the Primary Service.

`SwitchWindowEndTime` - Point on a metadata timeline at which an L0 switch may not occur. In the case where the service is selected at a point past the switchTimeStart, this value indicates the point when the switch should no longer occur.

`ExpirationTime` - The time at which the instruction is no longer valid and SHALL NOT cause a switch to occur. The switch engine MAY remove the instruction at this time. The time is num seconds from receipt by switch engine of instruction.

`InsertionIndivisible` - If true and the first trigger in an Insertion Group is missed then subsequent triggers in the Group SHALL be ignored. Used in conjunction with the GroupSize and PositionInGroup attributes.

`TriggerSwitch` - boolean with the following meanings:

- If TRUE - DPI Trigger initiates each switch.
- If FALSE - Metadata time initiates each switch.

NOTE: If TriggerSwitch = FALSE and time is past the SwitchStartTime when the SwitchInstruction is loaded the SwitchInstruction is executed immediately. This assumes that the switch engine is tuned to a service and has established a valid metadata timeline.

`InsertionOptions` - Options for insertion of content.

- GO_TO_BLACK - Go to black at SwitchEndTime even if insert program is not complete.
- RETURN_TO_PROGRAM - Return to Program at SwitchEndTime even if inserted content is not complete.
- PLAY_TO_COMPLETION - Play inserted program to completion under all circumstances.

`EndOfInsertion` - Identifies if the Switch Instruction should return to the Primary Service. Locator is a don't care.

**Throws:**

`java.lang.IllegalArgumentException` - Negative SourceID

`java.lang.IllegalArgumentException` - ExpirationTime passed

`java.lang.IllegalArgumentException` - Unknown InsertionOptions

`java.lang.IllegalArgumentException` - SwitchEndTime is before SwitchStartTime

`java.lang.IllegalArgumentException` - Duplicate SwitchID and SourceID.
`InvalidLocator` - Locator is invalid
`InvalidLocator` - Locator is null
`InvalidLocator` - Locator not found

# Method Detail

### setSwitchID

`public void` **`setSwitchID`**`(int SwitchID)`

Set attribute SwitchID to specified parameter.

**Parameters:**

`SwitchID` - Switch Identifier.

**Throws:**

`java.lang.IllegalStateException` - - if this method is called when the SwitchInstruction is Active (Has been added with the addInstruction).

### getSwitchID

`public int` **`getSwitchID`**`()`

Retrieve the attribute SwitchID.

**Returns:**

SwitchID

### setGroupSize

`public void` **`setGroupSize`**`(byte GroupSize)`

Set attribute GroupSize to specified parameter.

**Parameters:**

`GroupSize` - Number of Switches within a group.

**Throws:**

`java.lang.IllegalStateException` - - if this method is called when the SwitchInstruction is Active (Has been added with the addInstruction).

### getGroupSize

`public byte` **`getGroupSize`**`()`

Retrieve the attribute GroupSize.

**Returns:**

GroupSize

### setPositionInGroup

`public void` **`setPositionInGroup`**`(byte PositionInGroup)`

Set attribute PositionInGroup to specified parameter.

**Parameters:**

`PositionInGroup` - Position of Switch within a group.

**Throws:**

`java.lang.IllegalStateException` - - if this method is called when the SwitchInstruction is Active (Has been added with the addInstruction).

#### getPositionInGroup

`public byte` **`getPositionInGroup`**`()`

Retrieve the attribute PositionInGroup.

**Returns:**

PositionInGroup

#### setSourceID

`public void` **`setSourceID`**`(int SourceID)`

Set attribute SourceID to specified parameter.

**Parameters:**

`SourceID` - source identifier.

**Throws:**

`java.lang.IllegalStateException` - - if this method is called when the SwitchInstruction is Active (Has been added with the addInstruction).

#### getSourceID

`public int` **`getSourceID`**`()`

Retrieve the attribute SourceID.

**Returns:**

SourceID

#### setInsertionIndivisible

`public void` **`setInsertionIndivisible`**`(boolean InsertionIndivisible)`

Set attribute InsertionIndivisible to True or False.

**Parameters:**

`InsertionIndivisible` - Indicator of whether a partial Insertion Period can be processed.

**Throws:**

`java.lang.IllegalStateException` - - if this method is called when the SwitchInstruction is Active (Has been added with the addInstruction).

#### getInsertionIndivisible

`public boolean` **`getInsertionIndivisible`**`()`

Retrieve the attribute InsertionIndivisible.

**Returns:**

InsertionIndivisible

#### setTriggerSwitch

`public void` **`setTriggerSwitch`**`(boolean TriggerSwitch)`

Set attribute TriggerSwitch.

**Parameters:**

`TriggerSwitch` - boolean with the following meanings:

- TRUE - DPI Trigger initiated Break
- FALSE - Metadata time initiated Break

**Throws:**
`java.lang.IllegalStateException` - - if this method is called when the SwitchInstruction is Active (Has been added with the addInstruction).

#### getTriggerSwitch

```
public boolean getTriggerSwitch()
```

Get attribute TriggerSwitch.
**Returns:**
TriggerSwitch

#### setSwitchStartTime

```
public void setSwitchStartTime(int SwitchStartTime)
```

Set attribute SwitchStartTime to specified parameter.
**Parameters:**
`SwitchStartTime` - - Time insertion is to begin.
**Throws:**
`java.lang.IllegalStateException` - - if this method is called when the SwitchInstruction is Active (Has been added with the addInstruction).

#### getSwitchStartTime

```
public int getSwitchStartTime()
```

Retrieve the attribute SwitchStartTime.
**Returns:**
SwitchStartTime

#### setSwitchEndTime

```
public void setSwitchEndTime(int SwitchEndTime)
```

Set attribute SwitchEndTime to specified parameter.
**Parameters:**
`SwitchEndTime` - End of Insertion Period
**Throws:**
`java.lang.IllegalStateException` - - if this method is called when the SwitchInstruction is Active (Has been added with the addInstruction).

#### getSwitchEndTime

```
public int getSwitchEndTime()
```

Retrieve the attribute SwitchEndTime.
**Returns:**
SwitchEndTime

#### setExpirationTime

`public void` **`setExpirationTime`**`(int ExpirationTime)`

Set attribute ExpirationTime to specified parameter.

**Parameters:**

`ExpirationTime` - - The time when the Switch Instruction is no longer valid for an Insertion and can be removed.

**Throws:**

`java.lang.IllegalStateException` - - if this method is called when the SwitchInstruction is Active (Has been added with the addInstruction).

#### getExpirationTime

`public int` **`getExpirationTime`**`()`

Retrieve the attribute ExpirationTime.

**Returns:**

ExpirationTime

#### setSwitchWindowEndTime

`public void` **`setSwitchWindowEndTime`**`(int SwitchWindowEndTime)`

Set attribute SwitchWindowEndTime to specified parameter.

**Parameters:**

`SwitchWindowEndTime` - point on a metadata timeline at which an L0 switch may not occur. In the case where the service is selected at a point past the switchStartTime, this value indicates the point when the switch should no longer occur.

**Throws:**

`java.lang.IllegalStateException` - - if this method is called when the SwitchInstruction is Active (Has been added with the addInstruction).

#### getSwitchWindowEndTime

`public int` **`getSwitchWindowEndTime`**`()`

Retrieve the attribute SwitchWindowEndTime.

**Returns:**

SwitchWindowEndTime

#### setInsertionOptions

`public void` **`setInsertionOptions`**`(int InsertionOptions)`

Set attribute InsertionOptions to specified parameter.

**Parameters:**

`InsertionOptions` - Options for insertion of content.

- GO_TO_BLACK - Go to BLACK at SwitchEndTime even if insert program is not complete.
- RETURN_TO_PROGRAM - Return to Program at SwitchEndTime even if insert program is not complete.
- PLAY_TO_COMPLETION - Play inserted program to completion under all circumstances.

**Throws:**

`java.lang.IllegalStateException` - - if this method is called when the SwitchInstruction is Active (Has been added with the addInstruction).

#### getInsertionOptions

`public int` **`getInsertionOptions`**`()`

Retrieve the attribute InsertionOptions.

**Returns:**

InsertionOptions

#### setLocator

`public void` **`setLocator`**`(javax.tv.locator.Locator[] Locator)`

Set attribute Locator to specified parameter.

**Parameters:**

`Locator` - Identifies the component or program for insertion. Locator pointing to media content to switch to.

**Throws:**

`java.lang.IllegalStateException` - - if this method is called when the SwitchInstruction is Active (Has been added with the addInstruction).

#### getLocator

`public javax.tv.locator.Locator[]` **`getLocator`**`()`

Retrieve the attribute Locator.

**Returns:**

javax.tv.locator.Locator[]

#### setEndOfInsertion

`public void` **`setEndOfInsertion`**`(boolean EndOfInsertion)`

Set attribute EndOfInsertion to specified parameter.

**Parameters:**

`EndOfInsertion` - Identifies this as an end of Insertion Swtich Instruction that returns to the primary service.

**Throws:**

`java.lang.IllegalStateException` - - if this method is called when the SwitchInstruction is Active (Has been added with the addInstruction).

#### getEndOfInsertion

`public byte[]` **`getEndOfInsertion`**`()`

Retrieve the attribute EndOfInsertion.

**Returns:**

EndOfInsertion

# Annex Y OCAP 1.1 Environment API

## Package org.ocap.environment

| Interface Summary | |
|---|---|
| **EnvironmentListener** | The listener interface for receiving environment events. |

| Class Summary | |
|---|---|
| **Environment** | Represents an environment that provides the context in which applications run. |
| **EnvironmentEvent** | The `EnvironmentEvent` class is used to notify applications of events relating to environments. |
| **EnvironmentState** | Defines the set of available states for an environment. |
| **EnvironmentStateChangedEvent** | The `EnvironmentStateChangedEvent` class indicates the completion of a state transition of an environment. |

### org.ocap.environment
### Class Environment

```
java.lang.Object
  └ org.ocap.environment.Environment

public abstract class Environment
extends java.lang.Object
```

Represents an environment that provides the context in which applications run.

#### *27.3.4.1 Environment state machine*

Environments SHALL be in one of four states; inactive, selected, presenting and background. These are defined as follows;

- Environments in the inactive state SHALL have no running applications at all.
- Environments in the selected state have all applications running to the maximum extent possible and able to interact with the end-user.
- Environments in the presenting state may have running applications which are visible to the end-user but these SHALL NOT be able to receive input from the remote control except for PiP application keys. This SHALL be used for applications in a PiP or PoP session on hosts supporting applications in those sessions.
- Environments in the background state may have running applications but these applications SHALL NOT be in the normal mode.

#### 27.3.4.2 Transitions from selected to any other state and back

When the OCAP environment leaves the selected state, all assumable modules provided by the cable MSO SHALL be disabled and the manufacturer original enabled. When the OCAP environment enters the selected state, assumable modules provided by the cable MSO that were disabled SHALL be re-enabled.

#### 27.3.4.3 Transitions from selected or presenting to background

When an environment changes state from either selected or presenting to background, the following SHALL apply:

- applications able to run in cross-environment mode SHALL be put in cross-environment mode by the implementation
- applications able to run in background mode SHALL be put in background mode by the implementation
- applications signaled as pause-able MAY be put in the paused state by the implementation
- all other applications SHALL be terminated
- The implementation SHALL hide the user interfaces of applications which are put in background or paused mode or which are terminated. HScene instances shall have their visibility set to false.

NOTE: Need to specify if any events are generated when this happens.

#### 27.3.4.4 Transitions from background to selected or presenting

When an environment changes state from background to either selected or presenting, the following SHALL apply:

- all auto-start unbound applications which were terminated due to their environment going into the background state SHALL be started
- all auto-start bound applications which were terminated due to their environment going into the background state SHALL be started if still signaled in a selected service
- all applications from the newly selected environment that are running in cross-environment or background mode SHALL be returned to normal mode and restrictions on them as a consequence of them running in those modes SHALL be lifted
- visible user interfaces of cross-environment applications whose environment becomes selected SHALL continue to remain visible
- the applications and policy in the newly selected environment are responsible for determining which of the applications in that environment should be the first to have focus. That application is then responsible for requesting focus.
- Any pause-able applications which were paused when this environment went into the background state and which are still paused shall be returned to the active state

## Constructor Summary

| | |
|---|---|
| protected | **Environment**()<br>Constructor for environments. |

# Method Summary

| | |
|---|---|
| void | **addEnvironmentListener**(EnvironmentListener l)<br>Add a listener for environment events. |
| void | **deselect**()<br>Request this environment cease being selected. |
| static Environment | **getHome**()<br>Return the calling applications home environment |
| EnvironmentState | **getState**()<br>Queries the state of this environment. |
| void | **removeEnvironmentListener**(EnvironmentListener l)<br>Remove a listener for environment events. |
| void | **select**()<br>Request this environment become selected. |

| **Methods inherited from class java.lang.Object** |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Constructor Detail

### Environment

protected **Environment**()

Constructor for environments. This is provided for the use of implementations or other specifications and is not to be used by applications.

# Method Detail

### getHome

public static Environment **getHome**()

Return the calling applications home environment

**Returns:**

an environment

### addEnvironmentListener

public void **addEnvironmentListener**(EnvironmentListener l)

Add a listener for environment events.

**Parameters:**

l - the listener to add

#### removeEnvironmentListener

`public void **removeEnvironmentListener**(EnvironmentListener l)`

Remove a listener for environment events.

**Parameters:**

`l` - the listener to remove

#### getState

`public EnvironmentState **getState**()`

Queries the state of this environment.

**Returns:**

the state of this environment

#### select

`public void **select**()`

Request this environment become selected. This call is asynchronous and completion SHALL be reported with an EnvironmentEvent being sent to registered EnvironmentListeners.

This request SHALL be unconditionally granted except under the following circumstances.

- if a deadlock is detected with two or more environments repeatedly requesting they be selected each time they become de-selected. Implementations MAY include logic to detect this situation if it happens and refuse to change selected environment after an implementation specific number of changes in an implementation specific period.

- if this environment is in the presenting state due to it running in a PiP or PoP session and making this environment selected is not permitted by a PiP control mechanism on the OCAP host device

**Throws:**

`java.lang.IllegalStateException` - if a state change is already in progress for this environment or if the request fails for one of the circumstances defined above

`java.lang.SecurityException` - if and only if the calling application does not have `MonitorAppPermission("environment.selection")`

#### deselect

`public void **deselect**()`

Request this environment cease being selected.

NOTE It is implementation dependent which environment becomes selected when this call is used.

**Throws:**

`java.lang.SecurityException` - if and only if the calling application does not have `MonitorAppPermission("environment.selection")`

**org.ocap.environment**

## Class EnvironmentEvent

```
java.lang.Object
  └ java.util.EventObject
      └ org.ocap.environment.EnvironmentEvent
```

**All Implemented Interfaces:**

java.io.Serializable

**Direct Known Subclasses:**

EnvironmentStateChangedEvent

```
public abstract class EnvironmentEvent
extends java.util.EventObject
```

The `EnvironmentEvent` class is used to notify applications of events relating to environments.

# Field Summary

| Fields inherited from class java.util.EventObject |
|---|
| `source` |

# Constructor Summary

| |
|---|
| **`EnvironmentEvent`**`(Environment source)`<br>Simple constructor for these events. |

# Method Summary

| Methods inherited from class java.util.EventObject |
|---|
| `getSource, toString` |

| Methods inherited from class java.lang.Object |
|---|
| `clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait` |

# Constructor Detail

**EnvironmentEvent**

`public` **`EnvironmentEvent`**`(Environment source)`

Simple constructor for these events.

**Parameters:**

`source` - the environment which is the source of this event.

**org.ocap.environment**

## Interface EnvironmentListener

```
public interface EnvironmentListener
```

The listener interface for receiving environment events.

# Method Summary

| | |
|---|---|
| void | **environmentStateChanged**(EnvironmentEvent e)<br>Invoked when an application is to be notified of an event relating to an environment |

# Method Detail

**environmentStateChanged**

```
void environmentStateChanged(EnvironmentEvent e)
```

Invoked when an application is to be notified of an event relating to an environment

**Parameters:**

e - the event

**org.ocap.environment**
## Class EnvironmentState

```
java.lang.Object
  └ org.ocap.environment.EnvironmentState
```

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

Defines the set of available states for an environment.

# Field Summary

| | |
|---|---|
| `static EnvironmentState` | **`BACKGROUND`**<br>The environment is in the background. |
| `static EnvironmentState` | **`INACTIVE`**<br>The environment is inactive. |
| `static EnvironmentState` | **`PRESENTING`**<br>The environment is presenting. |
| `static EnvironmentState` | **`SELECTED`**<br>The environment is selected. |

# Constructor Summary

| | |
|---|---|
| `protected` | **`EnvironmentState`**`(java.lang.String s)`<br>This protected constructor is provided to enable the set of states to be extended. |

# Method Summary

| **Methods inherited from class java.lang.Object** |
|---|
| `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait` |

# Field Detail

### INACTIVE

```
public static final EnvironmentState INACTIVE
```

The environment is inactive. No applications are running.

#### SELECTED

`public static final EnvironmentState` **`SELECTED`**

The environment is selected. Applications are running to the maximum extent possible and are able to interact with the end-user.

#### PRESENTING

`public static final EnvironmentState` **`PRESENTING`**

The environment is presenting. Applications are running and can be visible to the end user. They cannot receive user input from the remote control except for PiP application keys.

#### BACKGROUND

`public static final EnvironmentState` **`BACKGROUND`**

The environment is in the background. Any running applications cannot be in the normal mode.

## Constructor Detail

#### EnvironmentState

`protected` **`EnvironmentState`**`(java.lang.String s)`

This protected constructor is provided to enable the set of states to be extended. It is not intended to be used by applications.

**org.ocap.environment**

## Class EnvironmentStateChangedEvent

```
java.lang.Object
  └ java.util.EventObject
      └ org.ocap.environment.EnvironmentEvent
          └ org.ocap.environment.EnvironmentStateChangedEvent
```

**All Implemented Interfaces:**

java.io.Serializable

```
public class EnvironmentStateChangedEvent
extends EnvironmentEvent
```

The `EnvironmentStateChangedEvent` class indicates the completion of a state transition of an environment. The following steps SHALL happen before a change of selected environment is reported as completed:

- the resource policy SHALL be changed to that of the new selected environment
- applications in the environment not allowed to run in its new state SHALL have been killed
- applications in the environment allowed to run but not allowed to show a user interface in its new state SHALL have that UI hidden, for OCAP-J applications this means the HScene SHALL be hidden with the same result as a call to setVisible(false).
- reservations of application exclusive events for the old applications which are allowed to run in the new selected environment SHALL have been cancelled
- if the newly selected environment has previously been in the selected or presenting state then all applications terminated when that environment last entered the background state have been re-started (where re-started for Xlets mean the call to the initXlet method has completed)
- returning to normal mode any running applications which are in cross-environment mode, background mode or paused mode
- returning to the active state any pause-able applications which were paused when this environment last stopped being selected and which are still paused

Reporting a change of selected environment as having been completed SHALL NOT wait for the following steps:

- completion of the re-starting of applications (where completion for Xlets means completion of calls to the startXlet method)
- requesting HScenes be visible
- requesting focus

When any screen re-draws happen is implementation dependent and may be deferred until the new applications are ready to redraw themselves.

# Field Summary

| Fields inherited from class java.util.EventObject |
|---|
| source |

# Constructor Summary

| |
|---|
| **EnvironmentStateChangedEvent**(Environment source, EnvironmentState fromstate, EnvironmentState tostate)<br>Create an EnvironmentStateChangedEvent object. |
| **EnvironmentStateChangedEvent**(Environment source, EnvironmentState fromstate, EnvironmentState tostate, org.dvb.application.AppID initialApplication)<br>Create an EnvironmentStateChangedEvent object specifying an initial application to run in the new environment. |

# Method Summary

| | |
|---|---|
| EnvironmentState | **getFromState**()<br>Return the state the environment was in before the state transition was requested as passed to the constructor of this event. |
| org.dvb.application.AppID | **getInitialApplication**()<br>Return initial application to run in the newly selected environment. |
| EnvironmentState | **getToState**()<br>Return the state the environment is in after the completion of the state transition as passed to the constructor of this event. |

| Methods inherited from class java.util.EventObject |
|---|
| getSource, toString |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

# Constructor Detail

### EnvironmentStateChangedEvent

```
public EnvironmentStateChangedEvent(Environment source,
                                    EnvironmentState fromstate,
                                    EnvironmentState tostate)
```

Create an EnvironmentStateChangedEvent object.

**Parameters:**

`source` - the `Environment` where the state transition happened

`fromstate` - the state the environment was in before the state transition * was requested

`tostate` - the state the environment is in after the completion of the state transition

### EnvironmentStateChangedEvent

```
public EnvironmentStateChangedEvent(Environment source,
                                    EnvironmentState fromstate,
                                    EnvironmentState tostate,
org.dvb.application.AppID initialApplication)
```

Create an EnvironmentStateChangedEvent object specifying an initial application to run in the new environment.

**Parameters:**

`source` - the `Environment` where the state transition happened

`fromstate` - the state the environment was in before the state transition was requested

`tostate` - the state the environment is in after the completion of the state transition

`initialApplication` - the initial application to run in the new environment

# Method Detail

### getFromState

```
public EnvironmentState getFromState()
```

Return the state the environment was in before the state transition was requested as passed to the constructor of this event.

**Returns:**

the old state

### getToState

```
public EnvironmentState getToState()
```

Return the state the environment is in after the completion of the state transition as passed to the constructor of this event.

**Returns:**

the new state

### getInitialApplication

```
public org.dvb.application.AppID getInitialApplication()
```

Return initial application to run in the newly selected environment.

**Returns:**

the initial application as passed to the constructor of this event or null if one was not provided

# Annex Z OCAP 1.1 Diagnostics API

## Package org.ocap.diagnostics

| Interface Summary | |
|---|---|
| **MIBDefinition** | This interface represents a MIB object that exposes its data type. |
| **MIBListener** | This interface represents a listener that can be registered with the `MIBManager` in order to listen for requests to MIB object encodings. |
| **MIBObject** | The interface represents a MIB Object. |
| **SNMPRequest** | This interface represents an application request for SNMP check, get, or set of a specific MIB. |
| **SNMPResponse** | This interface represents a response to an implementation request to an application that has registered control over a specific MIB. |

| Class Summary | |
|---|---|
| **MIBManager** | The MIBManager class provides Management Information Base (MIB) access to the host's MIB. |

**org.ocap.diagnostics**
## Interface MIBDefinition

**All Superinterfaces:**
MIBObject

public interface **MIBDefinition**
extends MIBObject

This interface represents a MIB object that exposes its data type. See RFC 2578 for data type definition.

# Field Summary

| | |
|---|---|
| static int | **SNMP_TYPE_BITS**<br>The BITS construct. |
| static int | **SNMP_TYPE_COUNTER32**<br>Base type, application defined 32 bit counter. |
| static int | **SNMP_TYPE_COUNTER64**<br>Base type, application defined 64 bit counter. |
| static int | **SNMP_TYPE_GAUGE32**<br>Base type, application defined 32 bit gauge. |
| static int | **SNMP_TYPE_INTEGER**<br>Base type, built-in ASN.1 integer type. |
| static int | **SNMP_TYPE_INVALID**<br>Unrecognized type encountered. |
| static int | **SNMP_TYPE_IPADDRESS**<br>Base type, application defined IP address. |
| static int | **SNMP_TYPE_OBJECTID**<br>Base type, built-in ASN.1 OBJECT IDENTIFIER type. |
| static int | **SNMP_TYPE_OCTETSTRING**<br>Base type, built-in ASN.1 string type. |
| static int | **SNMP_TYPE_OPAQUE**<br>Base type, application defined opaque variable. |
| static int | **SNMP_TYPE_TIMETICKS**<br>Base type, application defined time ticks. |

# Method Summary

| | |
|---|---|
| int | **getDataType**()<br>Gets the SNMP data type of the MIB. |

| Methods inherited from interface org.ocap.diagnostics.MIBObject |
|---|
| getData, getOID |

# Field Detail

### SNMP_TYPE_INVALID

```
public static final int SNMP_TYPE_INVALID
```
Unrecognized type encountered. Not defined by RFC 2578.

### SNMP_TYPE_INTEGER

```
public static final int SNMP_TYPE_INTEGER
```
Base type, built-in ASN.1 integer type.

### SNMP_TYPE_BITS

```
public static final int SNMP_TYPE_BITS
```
The BITS construct.

### SNMP_TYPE_OCTETSTRING

```
public static final int SNMP_TYPE_OCTETSTRING
```
Base type, built-in ASN.1 string type.

### SNMP_TYPE_OBJECTID

```
public static final int SNMP_TYPE_OBJECTID
```
Base type, built-in ASN.1 OBJECT IDENTIFIER type.

### SNMP_TYPE_IPADDRESS

```
public static final int SNMP_TYPE_IPADDRESS
```
Base type, application defined IP address.

### SNMP_TYPE_COUNTER32

```
public static final int SNMP_TYPE_COUNTER32
```
Base type, application defined 32 bit counter.

### SNMP_TYPE_GAUGE32

```
public static final int SNMP_TYPE_GAUGE32
```
Base type, application defined 32 bit gauge.

### SNMP_TYPE_TIMETICKS

```
public static final int SNMP_TYPE_TIMETICKS
```

Base type, application defined time ticks.

#### SNMP_TYPE_OPAQUE

```
public static final int SNMP_TYPE_OPAQUE
```

Base type, application defined opaque variable.

#### SNMP_TYPE_COUNTER64

```
public static final int SNMP_TYPE_COUNTER64
```

Base type, application defined 64 bit counter.

## Method Detail

#### getDataType

```
public int getDataType()
```

Gets the SNMP data type of the MIB.

**Returns:**

An SNMP data type defined by constants in this interface.

**org.ocap.diagnostics**
## Interface MIBListener

public interface **MIBListener**

This interface represents a listener that can be registered with the `MIBManager` in order to listen for requests to MIB object encodings.

# Method Summary

| | |
|---|---|
| `SNMPResponse` | **`notifySNMPRequest`**`(SNMPRequest request)`<br>Notifies a listener when a MIB object it registered control over has been requested. |

# Method Detail

### notifySNMPRequest

`public SNMPResponse` **`notifySNMPRequest`**`(SNMPRequest request)`

Notifies a listener when a MIB object it registered control over has been requested.

**Parameters:**

`request` - The request object containing the request type and OID.

**Returns:**

A response with the requested MIB encoding or an error status.

## org.ocap.diagnostics
## Class MIBManager

```
java.lang.Object
  └ org.ocap.diagnostics.MIBManager
```

public abstract class **MIBManager**
extends java.lang.Object

The MIBManager class provides Management Information Base (MIB) access to the host's MIB. Applications may use the MIBManager to make MIB queries. In addition, applications can set a MIB Object that can be retrieved from the Host device using SNMP.

# Constructor Summary

| | |
|---|---|
| protected | **MIBManager**()<br>Protected constructor, no application access. |

# Method Summary

| | |
|---|---|
| static MIBManager | **getInstance**()<br>Gets the MIBManager. |
| abstract MIBDefinition[] | **queryMibs**(java.lang.String oid)<br>Makes a query for all MIB objects matching the oid parameter, as well as any descendants in the MIB tree. |
| abstract void | **registerOID**(java.lang.String oid, int access, int dataType, MIBListener listener)<br>Registers a MIB object by adding the OID and listener to the MIB tables. |
| abstract void | **unregisterOID**(java.lang.String oid)<br>Unregisters a previously registered OID if the OID was registered by the same application. |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Constructor Detail

### MIBManager

protected **MIBManager**()
Protected constructor, no application access.

# Method Detail

### getInstance

```
public static MIBManager getInstance()
```

Gets the MIBManager.

**Returns:**

The MIBManager.

**Throws:**

`java.lang.SecurityException` - if the calling application does not have MonitorAppPermission("diagnostics").

### registerOID

```
public abstract void registerOID(java.lang.String oid,
                                 int access,
                                 int dataType,
                                 MIBListener listener)
```

Registers a MIB object by adding the OID and listener to the MIB tables. OID must be unique and not pre-existing. OID must not include trailing ".0" for leaf items or instance ID for table items. (The same listener may be installed to be invoked for any number of OIDs, if desired.) Table OIDs may be the OID of the start of the table or a column within the table, depending upon whether the implementation desires handling all table values with a single listener, or have different listeners for each column.

**Parameters:**

`oid` - The Object Identifier of the MIB being registered. The format of the string is based on the format defined by RFC 2578 for OBJECT IDENTIFIER definition. Terms in the string are period delimited, e.g. "1.3.6.1.4.1".

`access` - Indicates allowed access to the MIB being registered. See access constants in the class for valid values.

`dataType` - The data type of the MIB being registered. See constants in `MIBDefinition` for valid values.

`listener` - Listener to the MIB being registered. MAY be null.

**Throws:**

`java.lang.IllegalArgumentException` - if oid is an invalid oid string, if oid is already installed, or any other parameter has an invalid value.

### unregisterOID

```
public abstract void unregisterOID(java.lang.String oid)
```

Unregisters a previously registered OID if the OID was registered by the same application.

**Parameters:**

`oid` - An object identifier that was passed to the registerOID method.

**Throws:**

`java.lang.IllegalArgumentException` - if parameter does not match an OID registered with the registerOID method.

### queryMibs

```
public abstract MIBDefinition[] queryMibs(java.lang.String oid)
```

Makes a query for all MIB objects matching the oid parameter, as well as any descendants in the MIB tree.

**Parameters:**

`oid` - The object identifier to search for. The format of the string is based on the format defined by RFC 2578 for OBJECT IDENTIFIER definition. Terms in the string are period delimited, e.g. "1.3.6.1.4.1".

**Returns:**

An array of MIB definitions. The array is odered by increasing value of OID with the lowest value in the first element of the array.

**org.ocap.diagnostics**

## Interface MIBObject

**All Known Subinterfaces:**
MIBDefinition

public interface **MIBObject**

The interface represents a MIB Object. It contains the oid, as well as the encoding of the object with value formats corresponding to the ASN.1 definition of the object.

# Method Summary

| | |
|---|---|
| byte[] | **getData**()<br>Gets the current MIB object encoding in byte array form. |
| java.lang.String | **getOID**()<br>Gets the MIB object identifier. |

# Method Detail

### getOID

```
public java.lang.String getOID()
```

Gets the MIB object identifier.

**Returns:**
Object identifier of this MIB object. The object ID SHALL be formatted as per RFC 1778 section 2.15.

### getData

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

Gets the current MIB object encoding in byte array form. The array is formatted according to the ASN.1 format of the MIB.

**Returns:**
A byte array representing the MIB object encoding.

**org.ocap.diagnostics**
## Interface SNMPRequest

public interface **SNMPRequest**

This interface represents an application request for SNMP check, get, or set of a specific MIB.

# Field Summary

| | |
|---|---|
| static int | **SNMP_CHECK_FOR_SET_REQUEST**<br>Used to validate a MIB value before doing a set. |
| static int | **SNMP_GET_NEXT_REQUEST**<br>Get data for the next OID beyond the one passed in. |
| static int | **SNMP_GET_REQUEST**<br>Get data for the exact OID passed in. |
| static int | **SNMP_SET_REQUEST**<br>Set (modify) the value in the MIB for an OID. |

# Method Summary

| | |
|---|---|
| java.lang.String | **getMIBObject**()<br>Gets the MIB object for the request. |
| int | **getRequestType**()<br>Gets the type for this request. |

# Field Detail

### SNMP_CHECK_FOR_SET_REQUEST

public static final int **SNMP_CHECK_FOR_SET_REQUEST**
Used to validate a MIB value before doing a set.

### SNMP_SET_REQUEST

public static final int **SNMP_SET_REQUEST**
Set (modify) the value in the MIB for an OID.

### SNMP_GET_REQUEST

public static final int **SNMP_GET_REQUEST**
Get data for the exact OID passed in.

### SNMP_GET_NEXT_REQUEST

`public static final int` **`SNMP_GET_NEXT_REQUEST`**
Get data for the next OID beyond the one passed in.

# Method Detail

### getRequestType

`public int` **`getRequestType`**`()`
Gets the type for this request.
**Returns:**
One of the request types defined in this interface.

### getMIBObject

`public java.lang.String` **`getMIBObject`**`()`
Gets the MIB object for the request.
**Returns:**
MIB object for the request.

**org.ocap.diagnostics**

## Interface SNMPResponse

public interface **SNMPResponse**

This interface represents a response to an implementation request to an application that has registered control over a specific MIB.

# Field Summary

| | |
|---|---|
| static int | **SNMP_REQUEST_GENERIC_ERROR**<br>Any error not covered by the other error types. |
| static int | **SNMP_REQUEST_NO_SUCH_NAME**<br>The OID could not be found or there is no OID to respond to in a get next request. |
| static int | **SNMP_REQUEST_SET_OUT_OF_RANGE**<br>A Check/Set value error occurred. |
| static int | **SNMP_REQUEST_SUCCESS**<br>The request completed successfully and the MIBObject contains valid contents. |

# Method Summary

| | |
|---|---|
| MIBObject | **getMIBObject**()<br>Gets the encoding of the MIB object associated with the OID in the request that caused this response. |
| int | **getStatus**()<br>Get the status of the response. |

# Field Detail

### SNMP_REQUEST_SUCCESS

public static final int **SNMP_REQUEST_SUCCESS**

The request completed successfully and the MIBObject contains valid contents.

### SNMP_REQUEST_NO_SUCH_NAME

public static final int **SNMP_REQUEST_NO_SUCH_NAME**

The OID could not be found or there is no OID to respond to in a get next request.

### SNMP_REQUEST_SET_OUT_OF_RANGE

public static final int **SNMP_REQUEST_SET_OUT_OF_RANGE**

A Check/Set value error occurred.

#### SNMP_REQUEST_GENERIC_ERROR

```
public static final int SNMP_REQUEST_GENERIC_ERROR
```

Any error not covered by the other error types.

# Method Detail

#### getStatus

```
public int getStatus()
```

Get the status of the response.

**Returns:**

One of the request constants defined in this interface.

#### getMIBObject

```
public MIBObject getMIBObject()
```

Gets the encoding of the MIB object associated with the OID in the request that caused this response.

**Returns:**

If the getStatus method returns anything other than SNMP_REQUEST_SUCCESS this method returns the MIB Object, otherwise it returns null.

## End of OCAP 1.1 Profile