Notice

This OpenCable specification 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 the 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 implied, regarding the accuracy, completeness, or fitness for a particular purpose of this document, or any document referenced herein.

If you, or the company or organization that you represent, has an existing agreement with CableLabs concerning the use of this document, or subsequently enters into another agreement with CableLabs concerning the use of this document (e.g., the OCAP Implementers Agreement), your rights and obligations with respect to this document, and the rights and obligations of the company or organization that you represent, SHALL be as set forth therein.

© Cable Television Laboratories, Inc. 2001-2013
DISCLAIMER

This document is published by Cable Television Laboratories, Inc. ("CableLabs®").

CableLabs reserves the right to revise this document for any reason including, but not limited to, changes in laws, regulations, or standards promulgated by various agencies; technological advances; or changes in equipment design, manufacturing techniques, or operating procedures described, or referred to, herein. CableLabs makes no representation or warranty, express or implied, with respect to the completeness, accuracy, or utility of the document or any information or opinion contained in the report. Any use or reliance on the information or opinion is at the risk of the user, and CableLabs shall not be liable for any damage or injury incurred by any person arising out of the completeness, accuracy, or utility of any information or opinion contained in the document.

This document is not to be construed to suggest that any affiliated company modify or change any of its products or procedures, nor does this document represent a commitment by CableLabs or any cable member to purchase any product whether or not it meets the described characteristics. Nothing contained herein shall be construed to confer any license or right to any intellectual property, whether or not the use of any information herein necessarily utilizes such intellectual property. This document is not to be construed as an endorsement of any product or company or as the adoption or promulgation of any guidelines, standards, or recommendations.
Document Status Sheet

<table>
<thead>
<tr>
<th>Document Control Number:</th>
<th>OC-SP-OCAP1.3.1-130530</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document Title</td>
<td>OpenCable Application Platform (OCAP)</td>
</tr>
<tr>
<td>Revision History:</td>
<td>1.1.1 - 6/12/09</td>
</tr>
<tr>
<td></td>
<td>1.1.2 - 9/30/09</td>
</tr>
<tr>
<td></td>
<td>1.1.3 - 6/3/10</td>
</tr>
<tr>
<td></td>
<td>1.1.5 - 2/24/11</td>
</tr>
<tr>
<td></td>
<td>1.2 - 5/12/11</td>
</tr>
<tr>
<td></td>
<td>1.2.1 - 1/12/12</td>
</tr>
<tr>
<td></td>
<td>1.2.2 - 2/24/12</td>
</tr>
<tr>
<td></td>
<td>1.2.3 - 5/31/12</td>
</tr>
<tr>
<td></td>
<td>1.3 - 4/18/13</td>
</tr>
<tr>
<td></td>
<td>1.3.1 - 5/30/13</td>
</tr>
<tr>
<td>Date:</td>
<td>May 30, 2013</td>
</tr>
<tr>
<td>Status:</td>
<td>Work in Progress</td>
</tr>
<tr>
<td></td>
<td>Draft</td>
</tr>
<tr>
<td></td>
<td>Issued</td>
</tr>
<tr>
<td></td>
<td>Closed</td>
</tr>
</tbody>
</table>

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:

CableLabs® is a registered trademark of Cable Television Laboratories, Inc. Other CableLabs marks are listed at http://www.cablelabs.com/certqual/trademarks. All other marks are the property of their respective owners.
## Contents

1 **SCOPE** .............................................................................................................................................. 1
   1.1 DVB-GEM AND DVB-MHP SPECIFICATION CORRESPONDENCE ......................................................... 1
   1.2 OCAP SPECIFIC REQUIREMENTS ........................................................................................................ 1
       1.2.1 OCAP Purpose ................................................................................................................................... 1
       1.2.2 OCAP Application Areas ................................................................................................................ 1

2 **REFERENCES** .................................................................................................................................... 6
   2.1 DVB-GEM AND DVB-MHP SPECIFICATION CORRESPONDENCE ......................................................... 6
   2.2 OCAP SPECIFIC REQUIREMENTS ........................................................................................................ 6
       2.2.1 Normative References .................................................................................................................... 6
       2.2.2 Informative References ................................................................................................................ 10

3 **GLOSSARY** ......................................................................................................................................... 12
   3.1 DVB-GEM AND DVB-MHP SPECIFICATION CORRESPONDENCE ....................................................... 12
   3.2 OCAP SPECIFIC REQUIREMENTS ...................................................................................................... 12
       3.2.1 Definitions ..................................................................................................................................... 12

4 **ACRONYMS** ...................................................................................................................................... 26
   4.1 DVB-GEM AND DVB-MHP SPECIFICATION CORRESPONDENCE ....................................................... 26
   4.2 OCAP SPECIFIC REQUIREMENTS ...................................................................................................... 26
       4.2.1 Acronyms ..................................................................................................................................... 26

5 **CONVENTIONS** .................................................................................................................................. 30
   5.1 DVB-GEM AND DVB-MHP SPECIFICATION CORRESPONDENCE ....................................................... 30
   5.2 OCAP SPECIFIC REQUIREMENTS ...................................................................................................... 30
       5.2.1 Specification Language .................................................................................................................. 30
       5.2.2 Organization of the OCAP Specification ....................................................................................... 31

6 **GEM AND MHP CORRESPONDENCE** ............................................................................................... 33
   6.1 GEM COMPLIANCE .............................................................................................................................. 33
       6.1.1 GEM Errata .................................................................................................................................... 33
       6.1.2 GEM References .......................................................................................................................... 34
   6.2 DVB-GEM AND DVB-MHP FULL COMPLIANCE (INFORMATIVE) ....................................................... 36

7 **BASIC ARCHITECTURE** ..................................................................................................................... 39
   7.1 DVB-GEM AND DVB-MHP SPECIFICATION CORRESPONDENCE ....................................................... 39
   7.2 OCAP SPECIFIC REQUIREMENTS ...................................................................................................... 39
       7.2.1 Deviations from DVB-MHP ........................................................................................................... 40

8 **TRANSPORT PROTOCOLS** ................................................................................................................ 49
   8.1 DVB-GEM AND DVB-MHP SPECIFICATION CORRESPONDENCE ....................................................... 49
   8.2 OCAP SPECIFIC REQUIREMENTS ...................................................................................................... 51
       8.2.1 Deviations from DVB-MHP ........................................................................................................... 51
       8.2.2 Extensions to DVB-GEM (Normative) .......................................................................................... 53

9 **CONTENT FORMATS** .......................................................................................................................... 55
   9.1 DVB-GEM AND DVB-MHP SPECIFICATION CORRESPONDENCE ....................................................... 55
   9.2 OCAP SPECIFIC REQUIREMENTS ...................................................................................................... 56
       9.2.1 Deviations from the DVB-MHP Specification ................................................................................. 56
       9.2.2 Extensions to DVB-GEM .............................................................................................................. 57
9.2.3 OOB/DSG Object Carousel ................................................. 57
9.2.4 OpenType Font Support .................................................... 57

10 APPLICATION MODEL ............................................................ 58
10.1 DVB-GEM AND DVB-MHP SPECIFICATION CORRESPONDENCE ................................................................. 58
10.2 OCAP SPECIFIC REQUIREMENTS ........................................ 59
10.2.1 Deviations from the DVB-MHP Specification .................. 59
10.2.2 Extensions to DVB-GEM (Normative) .......................... 60

11 APPLICATION SIGNALING ..................................................... 83
11.1 DVB-GEM AND DVB-MHP SPECIFICATION CORRESPONDENCE ................................................................. 83
11.2 OCAP SPECIFIC REQUIREMENTS ........................................ 84
11.2.1 Deviations from the DVB-MHP Specification .................. 84
11.2.2 Extensions to DVB-GEM (Normative) ......................... 89

12 APPLICATION STORAGE ..................................................... 104
12.1 DVB-GEM AND DVB-MHP SPECIFICATION CORRESPONDENCE ................................................................. 104
12.2 OCAP REQUIREMENTS ........................................................ 104
12.2.1 Introduction ................................................................. 104
12.2.2 Storing Host device manufacturer applications ............. 104
12.2.3 MSO applications signaled in the XAIT ......................... 105
12.2.4 MSO applications installed through the Monitor Application ................................................................. 106
12.2.5 Populating the Services Database (Informative) .......... 107
12.2.6 Removal of Stored Applications on Change of MSO ...... 107
12.2.7 Authentication of Stored Applications ....................... 107
12.2.8 The Application description file .................................. 108
12.2.9 Partially-Stored Applications ........................................ 110

13 EXECUTION ENGINE PLATFORM ........................................ 111
13.1 DVB-GEM AND DVB-MHP SPECIFICATION CORRESPONDENCE (INFORMATIVE) ................................................................. 111
13.2 COMPLIANCE WITH GEM DVB-J PLATFORM ............... 111
13.2.1 Methods working on many locator types ..................... 111
13.3 OCAP EXTENSIONS TO GEM ............................................. 111
13.3.1 General Issues ............................................................. 112
13.3.2 OCAP Platform APIs .................................................... 113
13.3.3 OCAP Specific Network APIs ...................................... 115
13.3.4 Monitor Application Support ...................................... 115
13.3.5 Java Platform APIs ...................................................... 115
13.3.6 Presentation APIs ...................................................... 115
13.3.7 Streamed Media API ................................................... 117
13.3.8 Data Access APIs ..................................................... 118
13.3.9 Service Information and Selection APIs ..................... 124
13.3.10 Common Infrastructure APIs (informative) ................ 125
13.3.11 Termination of OCAP-J Applications by the Platform ................................................................. 126
13.3.12 Application Discovery and Launching APIs ............. 127
13.3.13 Security ................................................................. 127
13.3.14 Profile and Version Properties .................................. 127
13.3.15 Java Permissions ...................................................... 129
13.3.16 MHP 1.1 and OCAP .................................................. 130
13.3.17 JSR 280 XML Packages .......................................... 130
13.3.18 Logger API ............................................................ 131
13.3.19 System Logging ....................................................... 135
13.4 FULL JAVA API LIST (INFORMATIVE) .............................. 139
13.4.1 Java Platform Packages .............................................. 139
13.4.2 Java TV Packages ..................................................... 140
13.4.3 Java Media Framework 1.0 Packages ................................................................. 140
13.4.4 Java Secure Socket Extension 1.02 Packages .................................................. 140
13.4.5 HAVI Level 2 User Interface Packages ......................................................... 140
13.4.6 DVB-MHP 1.0.3 Packages .............................................................................. 140
13.4.7 DAVIC 1.4.1, Part 9 Packages ...................................................................... 140
13.4.8 OCAP Packages .......................................................................................... 141
13.4.9 MHP 1.1 Packages ...................................................................................... 141
13.4.10 Logger Packages ....................................................................................... 141
13.5 GEM and OCAP ............................................................................................ 141
13.5.1 VK_TELETEXT Key ...................................................................................... 141
13.5.2 Factory Methods ....................................................................................... 141
13.5.3 Locale Support (informative) ...................................................................... 142
13.5.4 org.dvb.event Package ............................................................................. 142

14 SECURITY ............................................................................................................ 143
14.1 DVB-GEM and DVB-MHP Specification Correspondence .................................. 143
14.2 OCAP Specific Requirements ............................................................................ 147
14.2.1 Deviations from the DVB-MHP Specification ........................................... 147
14.2.2 Extensions to DVB-GEM (Normative) ...................................................... 153

15 GRAPHICS REFERENCE MODEL .................................................................. 160
15.1 DVB-GEM and DVB-MHP Specification Correspondence .................................. 160
15.2 OCAP Specific Requirements ............................................................................ 161
15.2.1 Deviations from the DVB-MHP Specification ........................................... 161
15.2.2 Extensions to DVB-GEM (Normative) ...................................................... 162

16 SYSTEM INTEGRATION ASPECTS .................................................................. 164
16.1 DVB-GEM and DVB-MHP Specification Correspondence .................................. 164
16.2 OCAP Specific Requirements ............................................................................ 164
16.2.1 Deviations from DVB-MHP ...................................................................... 164
16.2.2 Extensions to DVB (normative) ................................................................. 180

17 DETAILED PLATFORM PROFILE DEFINITION ........................................... 181
17.1 DVB-GEM and DVB-MHP Specification Correspondence .................................. 181
17.2 OCAP Specific Requirements ............................................................................ 181
17.2.1 Deviations from DVB-MHP ...................................................................... 182

18 REGISTRY OF CONSTANTS ............................................................................. 183
18.1 DVB-GEM and DVB-MHP Specification Correspondence .................................. 183
18.2 OCAP Specific Requirements ............................................................................ 183
18.2.1 Deviations from the DVB-MHP Specification ........................................... 183

19 RESOURCE MANAGEMENT ............................................................................. 197
19.1 DVB-GEM and DVB-MHP Specification Correspondence .................................. 197
19.2 OCAP Specific Requirements ............................................................................ 197
19.2.1 Normative ............................................................................................... 197

20 BASELINE FUNCTIONALITY ........................................................................... 204
20.1 DVB-GEM and DVB-MHP Specification Correspondence .................................. 204
20.2 OCAP Specific Requirements ............................................................................ 204
20.2.1 Boot Process ............................................................................................ 204
20.2.2 Conceptual Module Descriptions ............................................................. 208

21 MONITOR FUNCTIONALITY ............................................................................. 218
21.1 DVB-GEM and DVB-MHP Specification Correspondence .................................. 218
21.2 OCAP SPECIFIC REQUIREMENTS ................................................................. 218
  21.2.1 Extensions to DVB-GEM (Informative) ............................................. 218

22 OBJECT CAROUSEL .................................................................................. 234
  22.1 DVB-GEM and DVB-MHP SPECIFICATION CORRESPONDENCE ......................................................... 234
  22.2 OCAP SPECIFIC REQUIREMENTS ......................................................... 236
    22.2.1 Deviations from the DVB-MHP Specification ............................... 236
    22.2.2 Extensions to DVB-MHP ............................................................... 238

23 TEXT PRESENTATION ............................................................................... 243
  23.1 DVB-GEM and DVB-MHP SPECIFICATION CORRESPONDENCE ......................................................... 243
  23.2 OCAP SPECIFIC REQUIREMENTS ......................................................... 244
    23.2.1 Deviations from the DVB-MHP Specification ............................... 244

24 EXTENSIONS ......................................................................................... 246
  24.1 DVB-GEM and DVB-MHP SPECIFICATION CORRESPONDENCE ......................................................... 246
  24.2 OCAP SPECIFIC REQUIREMENTS ......................................................... 246
    24.2.1 Extensions to DVB-GEM (Normative) ............................................. 246

25 MINIMUM PLATFORM CAPABILITIES ................................................. 247
  25.1 DVB-GEM and DVB-MHP SPECIFICATION CORRESPONDENCE ......................................................... 247
  25.2 OCAP SPECIFIC REQUIREMENTS ......................................................... 248
    25.2.1 Deviations from the DVB-MHP Specification ............................... 248
    25.2.2 Extensions to DVB-GEM (Normative) ............................................. 254
    25.2.3 Extensions to DVB-GEM (Informative) ............................................. 257

ANNEX A MULTI-FUNCTION IMPLEMENTATION DETAILS ....................... 259
  A.1 OVERVIEW .......................................................................................... 259
  A.2 OPERATIONAL RULES ........................................................................ 259
    A.2.1 Cross-environment Applications ..................................................... 259
    A.2.2 Multi-Function Host Handling of Services ...................................... 260
    A.2.3 Access to OCAP applications ....................................................... 260

ANNEX B OCAP SPECIFIC NETWORK APIS .......................................... 262
  B.1 DVB-GEM AND DVB-MHP SPECIFICATION CORRESPONDENCE ......................................................... 262
  B.2 OCAP SPECIFIC REQUIREMENTS ......................................................... 262
    B.2.1 Deviations to DVB-MHP ............................................................... 262
    B.2.2 Extensions to DVB-MHP (Normative) ............................................. 263
    B.2.3 Java.net Usage ............................................................................ 263

ANNEX C PERMISSIONS ............................................................................. 264
  C.1 DVB-GEM AND DVB-MHP SPECIFICATION CORRESPONDENCE ......................................................... 264
  C.2 OCAP SPECIFIC REQUIREMENTS ......................................................... 264
    C.2.1 Extensions to DVB-MHP (Normative) ............................................. 264

ANNEX D HAVI LEVEL 2 USER INTERFACE ............................................ 265
  D.1 DVB-GEM AND DVB-MHP SPECIFICATION CORRESPONDENCE ......................................................... 265
  D.2 OCAP SPECIFIC REQUIREMENTS ......................................................... 265
    D.2.1 Deviations from the HAVi Specification ......................................... 265
    D.2.2 Extensions to DVB-MHP (Normative) ............................................. 266

ANNEX E OCAP UI AND UI EVENT APIS .............................................. 267
  PACKAGE org.ocap.ui ............................................................................. 267
  PACKAGE org.ocap.ui DESCRIPTION ......................................................... 267
  org.ocap.ui INTERFACE HSceneBinding .................................................. 268
ANNEX F  OCAP HARDWARE API ................................................................. 283
  PACKAGE org.ocap.hardware ................................................................. 283
  PACKAGE org.ocap.hardware Description .................................................. 283
  org.ocap.hardware CLASS CopyControl ................................................... 284
  org.ocap.hardware CLASS Host ................................................................. 286
  org.ocap.hardware INTERFACE IEEE1394Node ........................................ 292
  org.ocap.hardware INTERFACE PowerModeChangeListener ................. 294
  org.ocap.hardware CLASS VideoOutputPort .............................................. 295

ANNEX G  OCAP APPLICATION API ............................................................ 301
  PACKAGE org.ocap.application ............................................................... 301
  PACKAGE org.ocap.application Description .............................................. 301
  Application Registration ........................................................................... 301
  Application Information ............................................................................ 302
  XAIT Updating Management ..................................................................... 302
  Policy and Security Management .............................................................. 302
  org.ocap.application CLASS AppFilter ................................................. 305
  org.ocap.application INTERFACE AppFilterHandler ................................ 308
  org.ocap.application CLASS AppManagerProxy ...................................... 309
  org.ocap.application CLASS AppPattern ............................................... 315
  org.ocap.application INTERFACE AppSignalHandler ......................... 319
  org.ocap.application INTERFACE OcapAppAttributes ......................... 320
  org.ocap.application CLASS OcapXcPermission ...................................... 325
  org.ocap.application CLASS PermissionInformation ............................ 330
  org.ocap.application INTERFACE SecurityPolicyHandler ...................... 333

ANNEX H  OCAP MPEG COMPONENT API ..................................................... 335
  PACKAGE org.ocap.mpeg ........................................................................ 335
  PACKAGE org.ocap.mpeg Description ...................................................... 335
  org.ocap.mpeg CLASS PODExtendedChannel .......................................... 336

ANNEX I  OCAP NET API ........................................................................... 338
  PACKAGE org.ocap.net ........................................................................... 338
  PACKAGE org.ocap.net Description .......................................................... 338
  org.ocap.net CLASS OcapLocator ............................................................. 339
  org.ocap.net CLASS OCRLocator ............................................................. 355
  org.ocap.net CLASS URLLocator ............................................................. 357

ANNEX J  DATAGRAM SOCKET BUFFER CONTROL ..................................... 359
  J.1  DVB-GEM and DVB-MHP Specification Correspondence .................. 359
  J.2  OCAP Specific Requirements ............................................................. 359
    J.2.1  Extensions to DVB/MHP (Normative) ............................................. 359

ANNEX K  OCAP EVENT API ..................................................................... 360
  K.1  DVB-GEM and DVB-MHP Specification Correspondence .................. 361
  K.2  OCAP Specific Requirements ............................................................ 361
    K.2.1  Event Filtering .............................................................................. 361
    K.2.2  Event Distribution ........................................................................ 361
    K.2.3  Event Resource Management ....................................................... 363
**ANEX R**  **OCAP HARDWARE POD API** .................................................................................................................. 448

```
PACKAGE ORG.OCAP.HARDWARE.POD.................................................................................................................. 448
PACKAGE ORG.OCAP.HARDWARE.POD DESCRIPTION ..................................................................................... 448
ORG.OCAP.HARDWARE.POD INTERFACE PodApplication .................................................................................. 456
ORG.OCAP.HARDWARE.POD CLASS Pod ............................................................................................................. 452
```

**ANEX S**  **OCAP MEDIA API** ........................................................................................................................... 459

```
PACKAGE ORG.OCAP.MEDIA.................................................................................................................................... 459
PACKAGE ORG.OCAP.MEDIA DESCRIPTION ......................................................................................................... 460
ORG.OCAP.MEDIA CLASS AlternativeMediaPresentationEvent .............................................................................. 461
ORG.OCAP.MEDIA INTERFACE AlternativeMediaPresentationReason ................................................................. 464
ORG.OCAP.MEDIA CLASS ClosedCaptioningAttribute .......................................................................................... 466
ORG.OCAP.MEDIA INTERFACE ClosedCaptioningControl .................................................................................... 475
ORG.OCAP.MEDIA CLASS ClosedCaptioningEvent ............................................................................................... 481
ORG.OCAP.MEDIA INTERFACE ClosedCaptioningListener ..................................................................................... 484
ORG.OCAP.MEDIA CLASS FilterResourceAvailableEvent ...................................................................................... 485
ORG.OCAP.MEDIA CLASS ForcedDisconnectedEvent .......................................................................................... 487
ORG.OCAP.MEDIA INTERFACE MediaAccessAuthorization .................................................................................... 489
ORG.OCAP.MEDIA INTERFACE MediaAccessConditionControl ............................................................................ 491
ORG.OCAP.MEDIA INTERFACE MediaAccessHandler ............................................................................................ 492
ORG.OCAP.MEDIA CLASS MediaAccessHandlerRegistrar ...................................................................................... 494
ORG.OCAP.MEDIA CLASS MediaPresentationEvaluationTrigger ............................................................................. 496
ORG.OCAP.MEDIA CLASS MediaPresentationEvent .................................................................................................. 499
ORG.OCAP.MEDIA CLASS MediaTimer .................................................................................................................... 502
ORG.OCAP.MEDIA INTERFACE MediaTimerListener .................................................................................................. 505
ORG.OCAP.MEDIA CLASS NormalMediaPresentationEvent ...................................................................................... 507
ORG.OCAP.MEDIA INTERFACE NotPresentedMediaInterface .................................................................................... 509
ORG.OCAP.MEDIA INTERFACE S3DConfiguration .................................................................................................. 510
ORG.OCAP.MEDIA INTERFACE S3DFormatTypes ..................................................................................................... 512
ORG.OCAP.MEDIA CLASS S3DSignalingChangedEvent .......................................................................................... 513
ORG.OCAP.MEDIA INTERFACE VBIFilter .................................................................................................................. 516
ORG.OCAP.MEDIA CLASS VBIFilterEvent .................................................................................................................. 520
ORG.OCAP.MEDIA CLASS VBIFilterGroup .................................................................................................................. 523
ORG.OCAP.MEDIA INTERFACE VBIFilterListener ..................................................................................................... 532
ORG.OCAP.MEDIA INTERFACE VideoFormatControl .............................................................................................. 533
```

**ANEX T**  **OCAP SI ACCESS API** ........................................................................................................................... 535

```
T.1  DVB-GEM and DVB_MHP Specification Correspondence .................................................................................. 535
T.2  OCAP Specific Requirements ............................................................................................................................. 535
  T.2.1  Extensions to DVB-MHP (Normative) ............................................................................................................ 535
  T.2.2  Mapping of the JavaTV SI API to Open Cable SI .......................................................................................... 537
```

---

EXAMPLE OF SETTING EAS AUDIO AND DISPLAY ATTRIBUTES ................................................................................. 415
ORG.OCAP.SYSTEM CLASS EASEvent .......................................................................................................................... 417
ORG.OCAP.SYSTEM INTERFACE EASHandler .................................................................................................................. 419
ORG.OCAP.SYSTEM INTERFACE EASListener .................................................................................................................. 420
ORG.OCAP.SYSTEM CLASS EASManager ....................................................................................................................... 421
ORG.OCAP.SYSTEM CLASS EASMODULERegistrar .......................................................................................................... 424
ORG.OCAP.SYSTEM CLASS MonitorAppPermission ....................................................................................................... 430
ORG.OCAP.SYSTEM CLASS RegisteredApiManager ...................................................................................................... 434
ORG.OCAP.SYSTEM CLASS RegisteredApiUserPermission .......................................................................................... 437
ORG.OCAP.SYSTEM INTERFACE SystemModule ............................................................................................................. 439
ORG.OCAP.SYSTEM INTERFACE SystemModuleHandler ............................................................................................... 441
ORG.OCAP.SYSTEM CLASS SystemModuleRegistrar ...................................................................................................... 443

---

CableLabs® 5/30/13
T.2.3 Mapping of the DAVIC SI API to OpenCable SI ........................................................... 547
PACKAGE org.ocap.si ............................................................................................................. 547
PACKAGE org.ocap.si DESCRIPTION .................................................................................... 548
org.ocap.si Class Descriptor .................................................................................................. 549
org.ocap.si INTERFACE DESCRIPTORTAG .......................................................................... 551
org.ocap.si INTERFACE PMTELEMENTARYSTREAMINFO ................................................. 561
org.ocap.si INTERFACE PROGRAMASSOCIATIONTABLE ................................................ 562
org.ocap.si Class PROGRAMASSOCIATIONTABLEMANAGER ........................................ 564
org.ocap.si INTERFACE PROGRAMMAPTABLE .................................................................... 566
org.ocap.si INTERFACE STREAMTYPE ................................................................................ 570
org.ocap.si INTERFACE REBOTEVENT ................................................................................. 572
org.ocap.si INTERFACE CABLECARDRESETEVENT ........................................................ 576
org.ocap.si INTERFACE RESOURCEDEPLETIONEVENT ..................................................... 579
org.ocap.si INTERFACE SYSTEMEVENT ........................................................................... 581
org.ocap.si INTERFACE SYSTEMEVENTLISTENER ............................................................ 584
org.ocap.si INTERFACE SYSTEMEVENTMANAGER ......................................................... 585

ANNEX U OCAP SYSTEM EVENT API .................................................................................... 585
PACKAGE org.ocap.system.event ......................................................................................... 585
PACKAGE org.ocap.system.event DESCRIPTION .................................................................. 585
org.ocap.system.event Class CABLECARDRESETEVENT ................................................. 587
org.ocap.system.event Class DEFERREDDOWNLOADEVENT ............................................. 589
org.ocap.system.event Class ERROREVENT ....................................................................... 591
org.ocap.system.event Class REBOTEVENT ...................................................................... 596
org.ocap.system.event Class RESOURCEDEPLETIONEVENT ......................................... 599
org.ocap.system.event Class SYSTEMEVENT .................................................................. 602
org.ocap.system.event INTERFACE SYSTEMEVENTLISTENER ....................................... 610
org.ocap.system.event INTERFACE SYSTEMEVENTMANAGER .................................. 611

ANNEX V OCAP STORAGE API .............................................................................................. 614
PACKAGE org.ocap.storage ................................................................................................. 614
PACKAGE org.ocap.storage DESCRIPTION ...................................................................... 614
org.ocap.storage INTERFACE AVAILABLESTORAGELISTENER ..................................... 615
org.ocap.storage INTERFACE DETACHABLESTORAGEOPTION .................................. 616
org.ocap.storage INTERFACE EXTENDEDFILEACCESSPERMISSIONS ......................... 618
org.ocap.storage INTERFACE LOGICALSTORAGEVOLUME .......................................... 621
org.ocap.storage INTERFACE REMOVABLESTORAGEOPTION .................................... 623
org.ocap.storage INTERFACE STORAGEMANAGER ...................................................... 624
org.ocap.storage INTERFACE STORAGEMANAGEREVENT ........................................... 627
org.ocap.storage INTERFACE STORAGEMANAGERLISTENER ..................................... 630
org.ocap.storage INTERFACE STORAGEOPTION ......................................................... 631
org.ocap.storage INTERFACE STORAGEPROXY ............................................................ 632

ANNEX W OCAP TEST API .................................................................................................... 638
PACKAGE org.ocap.test ....................................................................................................... 638
PACKAGE org.ocap.test DESCRIPTION ............................................................................. 638
org.ocap.test Class OCAPTest ............................................................................................ 647

ANNEX X (RESERVED) ........................................................................................................... 651

ANNEX Y ENVIRONMENT API ............................................................................................. 652
PACKAGE org.ocap.environment ......................................................................................... 652
PACKAGE org.ocap.environment DESCRIPTION ................................................................ 652
org.ocap.environment INTERFACE ENVIRONMENT .................................................... 653
ENVIRONMENT STATE MACHINE ....................................................................................... 653
TRANSITIONS FROM BACKGROUND TO SELECTED OR PRESENTING ............................. 653
TRANSITIONS FROM SELECTED OR PRESENTING TO BACKGROUND ................................. 653
APPENDIX I  REVISION HISTORY .............................................................................................................................. 758

Figures

Figure 7–1 - OCAP CONTEXT ........................................................................................................................................ 40
Figure 7–2 - OCAP SOFTWARE ARCHITECTURE ........................................................................................................ 41
Figure 11–1 - ADDRESSING DESCRIPTORS RELATIONSHIP ................................................................................... 96
Figure 11–2 - ADDRESSING DESCRIPTORS RELATIONSHIP EXAMPLE ............................................................................ 97
Figure 14–1 - CERTIFICATE CHAIN ILLUSTRATED ...................................................................................................... 151
Figure 16–1 - CA EVENT GENERATION FOR SERVICEContext.select(SERVICE) .......................................................... 177
Figure 16–2 - CA EVENT GENERATION FOR SERVICEContext.select(SERVICE) .......................................................... 178
Figure 20–1 - BOOT PROCESS FLOW CHART .................................................................................................................. 208
Figure 21–1 - OCAP APPLICATIONS ............................................................................................................................ 219
Figure 21–2 - REGISTERED API ACCESS PROCESS .................................................................................................... 227
Figure 25–1 - REQUIRED DEVICE RESOLUTIONS ....................................................................................................... 251
Figure K–1 - THE OCAP DOWNLOADED APPLICATION VISIBLE EVENT DISTRIBUTION MECHANISM .................. 362
Figure W–1 - CONNECTIVITY DIAGRAM ...................................................................................................................... 639
Figure W–2 - INTERACTION SUBSYSTEM DESIGN DIAGRAM ........................................................................................ 641

ANNEX Z  DIAGNOSTICS API............................................................................................................................................. 665

PACKAGE org.ocap.diagnostics ...................................................................................................................................... 665
org.ocap.diagnostics Interface MIBDefinition ................................................................................................................ 666
org.ocap.diagnostics Interface MIBListener .................................................................................................................. 669
org.ocap.diagnostics Class MIBManager .................................................................................................................... 670
org.ocap.diagnostics Class MIBObject ........................................................................................................................ 675
org.ocap.diagnostics Interface SNMPRequest ........................................................................................................... 677
org.ocap.diagnostics Class SNMPResponse ................................................................................................................ 679

ANNEX AA  LOGGER API..................................................................................................................................................... 684

PACKAGE org.ocap.logging ........................................................................................................................................... 684
org.ocap.logging Class MIBAppender ............................................................................................................................ 684
PACKAGE org.apache.log4j ........................................................................................................................................... 687
org.apache.log4j Class Category .................................................................................................................................. 687
org.apache.log4j Class Hierarchy ................................................................................................................................ 702
org.apache.log4j Class Logger ...................................................................................................................................... 709
org.apache.log4j Class LogManager .................................................................................................................................. 737
org.apache.log4j Class PropertyConfigurator ........................................................................................................... 740
PACKAGE org.apache.log4j.config ............................................................................................................................. 747
org.apache.log4j.config Class PropertySetter ........................................................................................................... 747
PACKAGE org.apache.log4j.spi ..................................................................................................................................... 749
org.apache.log4j.spi Interface LoggerRepository ..................................................................................................... 749
PACKAGE org.apache.log4j.xml ................................................................................................................................. 752
org.apache.log4j.xml Class DOMConfigurator ........................................................................................................... 752

ORG.APACHE.LOG4J.XML CLASS DOMCONFIGURATOR .......................................................................................... 752
PACKAGE ORG.APACHE.LOG4J.XML ........................................................................................................................ 752
ORG.APACHE.LOG4J.SPI INTERFACE LOGGERREPOSITORY ...................................................................................... 749
PACKAGE ORG.APACHE.LOG4J.SPI .............................................................................................................................. 749
ORG.APACHE.LOG4J CONFIG CLASS PROPERTYSETTER .................................................................................................. 747
PACKAGE ORG.APACHE.LOG4J.CONFIG ................................................................................................................... 747
ORG.APACHE.LOG4J CLASS PROPERTYCONFIGURATOR .......................................................................................... 740
ORG.APACHE.LOG4J CLASS LOGMANAGER ............................................................................................................. 737
ORG.APACHE.LOG4J CLASS LOGGER ....................................................................................................................... 709
ORG.APACHE.LOG4J CLASS CATEGORY ................................................................................................................... 687
PACKAGE ORG.APACHE.LOG4J ................................................................................................................................ 687
ORG.APACHE.LOG4J CLASS HIERARCHY ......................................................................................................................... 702
PACKAGE ORG.APACHE.LOG4J ................................................................................................................................ 687
ORG.APACHE.LOG4J CLASS CATEGORY ................................................................................................................... 687
PACKAGEn ORG.OCAP.LOGGING ................................................................................................................................ 684
ORG.OCAP.DIAGNOSTICS CLASS SNMPRESPONSE ................................................................................................. 679
ORG.OCAP.DIAGNOSTICS INTERFACE SNMPREQUEST ........................................................................................... 677
ORG.OCAP.DIAGNOSTICS CLASS MIBMANAGER- ................................................................................................... 670
ORG.OCAP.DIAGNOSTICS INTERFACE MIBLISTENER .............................................................................................. 669
ORG.OCAP.DIAGNOSTICS CLASS MIBOBJECT- ........................................................................................................ 675
ORG.OCAP.DIAGNOSTICS INTERFACE MIBDEFINITION ........................................................................................... 666
PACKAGE ORG.OCAP.DIAGNOSTICS ......................................................................................................................... 665
ORG.OCAP.ENVIRONMENT CLASS ENVIRONMENTSTATE .............................................................. 660
ORG.OCAP.ENVIRONMENT Interface ENVIRONMENTLISTENER ............................................................ 657
ORG.OCAP.ENVIRONMENT Class ENVIRONMENTEvent .......................................................................................... 657
ORG.OCAP.ENVIRONMENT Class ENVIRONMENTState ........................................................................................... 659
ORG.OCAP.ENVIRONMENT Class ENVIRONMENTStateChangedEvent ............................................................... 662

Figures
# Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE 1–1</td>
<td>CORRELATION BETWEEN OCAP, [DVB-GEM 1.0.2] AND [DVB-MHP 1.0.3]</td>
<td>1</td>
</tr>
<tr>
<td>TABLE 2–1</td>
<td>CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] AND [DVB-MHP 1.0.3]</td>
<td>6</td>
</tr>
<tr>
<td>TABLE 2–2</td>
<td>OCAP NORMATIVE REFERENCES</td>
<td>7</td>
</tr>
<tr>
<td>TABLE 2–3</td>
<td>OCAP INFORMATIVE REFERENCES</td>
<td>10</td>
</tr>
<tr>
<td>TABLE 3–1</td>
<td>CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] AND [DVB-MHP 1.0.3]</td>
<td>12</td>
</tr>
<tr>
<td>TABLE 4–1</td>
<td>CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] AND [DVB-MHP 1.0.3]</td>
<td>26</td>
</tr>
<tr>
<td>TABLE 5–1</td>
<td>CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2]</td>
<td>30</td>
</tr>
<tr>
<td>TABLE 6–1</td>
<td>APPLICATION AUTHENTICATION FUNCTIONAL EQUIVALENT</td>
<td>34</td>
</tr>
<tr>
<td>TABLE 6–2</td>
<td>COMPLIANCE TABLE</td>
<td>37</td>
</tr>
<tr>
<td>TABLE 7–1</td>
<td>CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] AND [DVB-MHP 1.0.3]</td>
<td>39</td>
</tr>
<tr>
<td>TABLE 8–1</td>
<td>CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] AND [DVB-MHP 1.0.3]</td>
<td>49</td>
</tr>
<tr>
<td>TABLE 9–1</td>
<td>CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] AND [DVB-MHP 1.0.3]</td>
<td>55</td>
</tr>
<tr>
<td>TABLE 9–2</td>
<td>EXTENSIONS TO [DVB-MHP 1.0.3] MIME TYPES</td>
<td>57</td>
</tr>
<tr>
<td>TABLE 10–1</td>
<td>CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] AND [DVB-MHP 1.0.3]</td>
<td>58</td>
</tr>
<tr>
<td>TABLE 10–2</td>
<td>COMBINATIONS OF ENVIRONMENTS AND STATES</td>
<td>76</td>
</tr>
<tr>
<td>TABLE 11–1</td>
<td>CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] AND [DVB-MHP 1.0.3]</td>
<td>79</td>
</tr>
<tr>
<td>TABLE 11–2</td>
<td>PROTOCOL_ID</td>
<td>86</td>
</tr>
<tr>
<td>TABLE 11–3</td>
<td>SEMANTIC OF SELECTOR BYTES</td>
<td>86</td>
</tr>
<tr>
<td>TABLE 11–4</td>
<td>SYNTAX OF SELECTOR BYTES FOR INTERACTION CHANNEL</td>
<td>87</td>
</tr>
<tr>
<td>TABLE 11–5</td>
<td>VALUE RANGES FOR APPLICATION_ID IN UNBOUND APPLICATIONS</td>
<td>90</td>
</tr>
<tr>
<td>TABLE 11–6</td>
<td>ABSTRACT SERVICE DESCRIPTOR</td>
<td>92</td>
</tr>
<tr>
<td>TABLE 11–7</td>
<td>SERVICE_ID VALUE RANGE</td>
<td>92</td>
</tr>
<tr>
<td>TABLE 11–8</td>
<td>UNBOUND APPLICATION DESCRIPTOR</td>
<td>93</td>
</tr>
<tr>
<td>TABLE 11–9</td>
<td>PRIVILEGED CERTIFICATE DESCRIPTOR</td>
<td>93</td>
</tr>
<tr>
<td>TABLE 11–10</td>
<td>APPLICATION STORAGE DESCRIPTOR</td>
<td>94</td>
</tr>
<tr>
<td>TABLE 11–11</td>
<td>APPLICATION MODE DESCRIPTOR</td>
<td>94</td>
</tr>
<tr>
<td>TABLE 11–12</td>
<td>VALUES FOR MODE VARIABLE OF APPLICATION MODE DESCRIPTOR</td>
<td>94</td>
</tr>
<tr>
<td>TABLE 11–13</td>
<td>REGISTERED API DESCRIPTOR</td>
<td>95</td>
</tr>
<tr>
<td>TABLE 11–14</td>
<td>ADDRESSING_DESCRIPTOR</td>
<td>98</td>
</tr>
<tr>
<td>TABLE 11–15</td>
<td>HOST_ATTRIBUTE_COMPARISON_OPERATION</td>
<td>99</td>
</tr>
<tr>
<td>TABLE 11–16</td>
<td>HOST ATTRIBUTE COMPARISON TYPES</td>
<td>99</td>
</tr>
<tr>
<td>TABLE 11–17</td>
<td>AND OPERATION</td>
<td>100</td>
</tr>
<tr>
<td>TABLE 11–18</td>
<td>OR OPERATION</td>
<td>100</td>
</tr>
<tr>
<td>TABLE 11–19</td>
<td>NOT OPERATION</td>
<td>100</td>
</tr>
<tr>
<td>TABLE 11–20</td>
<td>TRUE OPERATION</td>
<td>100</td>
</tr>
<tr>
<td>TABLE 11–21</td>
<td>ADDRESSING_ATTRIBUTE_DESCRIPTOR</td>
<td>101</td>
</tr>
<tr>
<td>TABLE 11–22</td>
<td>ADDRESSABLE_APPLICATION_DESCRIPTOR</td>
<td>101</td>
</tr>
<tr>
<td>TABLE 12–1</td>
<td>CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] AND [DVB-MHP 1.0.3]</td>
<td>104</td>
</tr>
<tr>
<td>TABLE 12–2</td>
<td>STORAGE APPLICATIONS</td>
<td>105</td>
</tr>
<tr>
<td>TABLE 13–1</td>
<td>CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2]</td>
<td>111</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>------</td>
<td>------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>14–1</td>
<td>Correlation between OCAP and [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3]</td>
<td>143</td>
</tr>
<tr>
<td>14–2</td>
<td>Attributes for an <a href="">ocap:ixc/</a> permission request file entry</td>
<td>159</td>
</tr>
<tr>
<td>15–1</td>
<td>Correlation between OCAP and [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3]</td>
<td>160</td>
</tr>
<tr>
<td>15–2</td>
<td>Typical resolutions and their pixel aspect ratio</td>
<td>162</td>
</tr>
<tr>
<td>16–1</td>
<td>Correlation between OCAP and [DVB-GEM 1.0.2]</td>
<td>164</td>
</tr>
<tr>
<td>16–2</td>
<td>OCAP URI universally resolvable constructs</td>
<td>169</td>
</tr>
<tr>
<td>16–3</td>
<td>OCAP URI environment specific constructs</td>
<td>169</td>
</tr>
<tr>
<td>16–4</td>
<td>OCAP URI physical layer constructs</td>
<td>170</td>
</tr>
<tr>
<td>16–5</td>
<td>OCAP URL examples</td>
<td>170</td>
</tr>
<tr>
<td>16–6</td>
<td>OCAP addressable entities</td>
<td>172</td>
</tr>
<tr>
<td>16–7</td>
<td>CA events</td>
<td>179</td>
</tr>
<tr>
<td>17–1</td>
<td>Correlation between OCAP and [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3]</td>
<td>181</td>
</tr>
<tr>
<td>18–1</td>
<td>Correlation between OCAP and [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3]</td>
<td>183</td>
</tr>
<tr>
<td>18–2</td>
<td>OCAP extension to Table 69 of [DVB-MHP 1.0.3]</td>
<td>184</td>
</tr>
<tr>
<td>19–1</td>
<td>Correlation between OCAP and [DVB-GEM 1.0.2]</td>
<td>197</td>
</tr>
<tr>
<td>20–1</td>
<td>Correlation between OCAP and [DVB-GEM 1.0.2]</td>
<td>204</td>
</tr>
<tr>
<td>21–1</td>
<td>Correlation between OCAP and [DVB-GEM 1.0.2]</td>
<td>218</td>
</tr>
<tr>
<td>21–2</td>
<td>Triggers</td>
<td>231</td>
</tr>
<tr>
<td>21–3</td>
<td>Reasons</td>
<td>232</td>
</tr>
<tr>
<td>21–4</td>
<td>OCAP-specific user preference settings</td>
<td>232</td>
</tr>
<tr>
<td>22–1</td>
<td>Correlation between OCAP and [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3]</td>
<td>234</td>
</tr>
<tr>
<td>22–2</td>
<td>OCAP nsap</td>
<td>237</td>
</tr>
<tr>
<td>22–3</td>
<td>org.dvb.dsmcc.serviceDomain errors while tuning to an analog source</td>
<td>241</td>
</tr>
<tr>
<td>22–4</td>
<td>org.dvb.dsmcc.dsmccoobject errors while tuning to an analog source</td>
<td>242</td>
</tr>
<tr>
<td>23–1</td>
<td>Correlation between OCAP and [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3]</td>
<td>243</td>
</tr>
<tr>
<td>24–1</td>
<td>Correlation between OCAP and [DVB-GEM 1.0.2]</td>
<td>246</td>
</tr>
<tr>
<td>25–1</td>
<td>Correlation between OCAP and [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3]</td>
<td>247</td>
</tr>
<tr>
<td>25–2</td>
<td>Required device resolutions - sets 1 and 2</td>
<td>249</td>
</tr>
<tr>
<td>25–3</td>
<td>Required device resolutions - sets 3 and 4</td>
<td>249</td>
</tr>
<tr>
<td>25–4</td>
<td>Required device resolutions - sets 5 through 8</td>
<td>249</td>
</tr>
<tr>
<td>25–5</td>
<td>Minimum keycode set</td>
<td>252</td>
</tr>
<tr>
<td>25–6</td>
<td>Minimum keyboard keycode set</td>
<td>253</td>
</tr>
<tr>
<td>25–7</td>
<td>Minimum requirements for other resources for conformance purposes</td>
<td>256</td>
</tr>
<tr>
<td>26–1</td>
<td>Correlation between OCAP and [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3]</td>
<td>262</td>
</tr>
<tr>
<td>26–2</td>
<td>Correlation between OCAP and [DVB-GEM 1.0.2]</td>
<td>264</td>
</tr>
<tr>
<td>26–3</td>
<td>Correlation between OCAP and [DVB-GEM 1.0.2]</td>
<td>265</td>
</tr>
<tr>
<td>26–4</td>
<td>Correlation between OCAP and [DVB-GEM 1.0.2]</td>
<td>267</td>
</tr>
<tr>
<td>26–5</td>
<td>Correlation between OCAP and [DVB-GEM 1.0.2]</td>
<td>283</td>
</tr>
<tr>
<td>26–6</td>
<td>Correlation between OCAP and [DVB-GEM 1.0.2]</td>
<td>301</td>
</tr>
<tr>
<td>26–7</td>
<td>Correlation between OCAP and [DVB-GEM 1.0.2]</td>
<td>335</td>
</tr>
</tbody>
</table>
TABLE I–1 - CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] ................................................................. 338
TABLE J–1 - CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] AND [DVB-MHP 1.0.3] .............................. 359
TABLE K–1 - CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] AND [DVB-MHP 1.0.3] .......................... 361
TABLE L–1 - CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] ............................................................. 376
TABLE M–1 - DVB-GEM SPECIFICATION COMPLIANCE ....................................................................................... 389
TABLE M–2 - CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] AND DVB-MHP 1.0 ............................... 389
TABLE N–1 - CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] AND [DVB-MHP 1.0.3] ......................... 393
TABLE O–1 - CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] AND [DVB-MHP 1.0.3] .................... 394
TABLE P–1 - CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] ........................................................... 397
TABLE Q–1 - CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] ........................................................... 413
TABLE R–1 - CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] ........................................................... 448
TABLE S–1 - CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] ........................................................... 459
TABLE T–1 - CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] AND [DVB-MHP 1.0.3] ......................... 535
TABLE T–2 - BEHAVIOR OF JAVA TV GETSERVICE() / SELECT() METHODS WITH REGARD TO OOB OCAP LOCATORS AND/OR OOB SI ............................................................... 537
TABLE T–3 - GETSERVICETYPE MAPPING ........................................................................................................... 539
TABLE T–4 - GETSTREAMTYPE MAPPING ........................................................................................................... 543
TABLE U–1 - CORRELATION BETWEEN OCAP AND [DVB-GEM 1.0.2] ............................................................. 585
This page intentionally left blank.
1 SCOPE

This section contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 1 Scope.

This document defines a minimum 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.

OCAP is based on the [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3] specification. Each section of OCAP contains a correspondence Table, indicating the corresponding [DVB-GEM 1.0.2] or [DVB-MHP 1.0.3] section and how OCAP complies with [DVB-GEM 1.0.2] or [DVB-MHP 1.0.3] section or deviates from the [DVB-MHP 1.0.3] sections. Whenever appropriate, OCAP directly references [DVB-GEM 1.0.2] or [DVB-MHP 1.0.3].

1.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

This section corresponds to [DVB-MHP 1.0.3] (there is no corresponding section in [DVB-GEM 1.0.2]) as follows:

| Table 1–1 - Correlation between OCAP, [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3] |
|---------------------------------|------------------------------|-----------------|
| OCAP                           | [DVB-GEM 1.0.2] Section      | GEM Compliance  |
| 1 Scope                        | 1 Scope                      | Extension       |
| 1.1 DVB-GEM and DVB-MHP        | No Corresponding Section     | OCAP-Specific Extension |
| Specification Correspondence   | No Corresponding Section     | OCAP-Specific Extension |
| 1.2 OCAP Specific Requirements | No Corresponding Section     | OCAP-Specific Extension |
| No Corresponding Section       | 1 Scope                      | Compliance      |

1.2 OCAP Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] section.

1.2.1 OCAP Purpose

OCAP 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 OCAP will be executed and/or displayed on OpenCable certified retail host devices. OCAP will allow cable operators, content providers, and CD manufacturers to deploy applications and services on all OpenCable-compliant host devices.

The OCAP 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 is to enable competing implementations of the OCAP interface for selection by the CE manufacturers.

1.2.2 OCAP Application Areas

1.2.2.1 Normative

The following information is normative for OCAP. It is specific to OCAP and is not contained within [DVB-MHP 1.0.3].
This specification has identified three classifications of applications which can run on an OCAP terminal:

- bound applications
- unbound applications
- non-OCAP 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-OCAP Applications

OpenCable recognizes the need for applications placed on a particular OCAP 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 Informative

The following information is informative to OCAP. It attempts to clarify issues in the [DVB-MHP 1.0.3], 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-compliant terminal. These applications will typically be implemented on top of OCAP as bound or unbound applications.

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

#### 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 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 Watching TV

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

1.2.2.2.1 SDTV

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

1.2.2.2.2 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.

1.2.2.2.3 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.

1.2.2.2.4 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 terminal and is supported by the OpenCable Application Platform.

1.2.2.2.5 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 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.

1.2.2.2.6 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.
1.2.2.2.7  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.

1.2.2.3  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.

1.2.2.4  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.

1.2.2.5  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.

1.2.2.5.1  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.

1.2.2.5.2  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.

1.2.2.6  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.

1.2.2.7  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.

1.2.2.9 Games

Some games will be implemented using OCAP. OCAP 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. "Twitch" games are an example. In this case, the application may have to use the native-code interface to specific OCAP terminals.

1.2.2.10 Music / Radio

Music broadcast in MPEG transport streams is another service potentially offered by the cable operators. OCAP 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.

1.2.2.11 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.

1.2.2.11.1 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.

1.2.2.11.2 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 a user decided to check the balance of their savings account while watching a basketball game, they should be able to.
2 REFERENCES

This section contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 2 References and are extensions of [DVB-MHP 1.0.3] 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 1.0.2] Section.

Table 2–1 - Correlation between OCAP and [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3]

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>MHP Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 References</td>
<td>2 References</td>
<td>Extension</td>
<td>2 References</td>
<td>Extension</td>
</tr>
<tr>
<td>2.1 DVB-GEM and DVB-MHP Specification</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>Correspondence</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.2 OCAP Specific Requirements</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>2.2.1 Normative References</td>
<td>Section 2, References</td>
<td>Extension</td>
<td>Section 2, References</td>
<td>Extension</td>
</tr>
<tr>
<td>2.2.2 Informative References</td>
<td>Annex C</td>
<td>Extension</td>
<td>Annex C</td>
<td>Extension</td>
</tr>
</tbody>
</table>

NOTE: OCAP indices in Table 2–2 do not have a one-to-one relationship with corresponding indices used in [DVB-MHP 1.0.3].

2.2 OCAP Specific Requirements

This subsection contains OCAP-specific requirements that does not correspond to any [DVB-GEM 1.0.2] Section.

2.2.1 Normative References

The following information is normative for the OCAP Specification. It is specific to OCAP and is not contained within [DVB-MHP 1.0.3].

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

All references are subject to revision, and parties to agreement based on this specification are encouraged to investigate the possibility of applying the most recent editions of the documents listed below.

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, non-Bundle reference, the latest version applies.
- For non-specific CableLabs references that are part of the [OC-BUNDLE], the versions mandated in a particular Bundle apply.
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-MHP 1.0.3]. 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.
8. ITU recommendations are available from ITU at http://www.itu.int for a fee.
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.
13. Java TV 1.1.1 supersedes the references to both Java TV 1.0 and JMF 1.0 in [DVB-MHP 1.0.3].
14. The OpenCable Bundle Requirements specification [OC-BUNDLE] indicates the set of CableLabs specifications required for the implementation of the OpenCable Bundle. The version number of [OC-BUNDLE] corresponds to the release number of the OpenCable Bundle that it describes. One or more versions of [OC-BUNDLE] reference this specification. Current and past versions of [OC-BUNDLE] may be obtained from CableLabs at http://www.cablelabs.com/opencable/specifications.

### Table 2–2 - OCAP Normative References

<table>
<thead>
<tr>
<th>References</th>
<th>Edition</th>
<th>Description</th>
<th>See Notes Section 2.2.1.1.1</th>
<th>MHP Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>References</td>
<td>Edition</td>
<td>Description</td>
<td>See Notes</td>
<td>MHP Index</td>
</tr>
<tr>
<td>---------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>[OC-BUNDLE]</td>
<td></td>
<td>OC-SP-BUNDLE, OpenCable Bundle Requirements</td>
<td>[11], [14]</td>
<td></td>
</tr>
<tr>
<td>[CCCP]</td>
<td></td>
<td>OpenCable CableCARD Copy Protection 2.0 Interface Specification, OC-SP-CCCP2.0, Cable Television Laboratories, Inc. Referenced in [OC-BUNDLE].</td>
<td></td>
<td>NA</td>
</tr>
<tr>
<td>[CCIF]</td>
<td></td>
<td>OpenCable CableCARD Interface 2.0 Specification, OC-SP-CCIF2.0, Cable Television Laboratories, Inc. Referenced in [OC-BUNDLE].</td>
<td></td>
<td>NA</td>
</tr>
<tr>
<td>[CDL]</td>
<td></td>
<td>OpenCable Common Download 2.0 Specification, OC-SP-CDL2.0, Cable Television Laboratories, Inc. Referenced in [OC-BUNDLE].</td>
<td></td>
<td>NA</td>
</tr>
<tr>
<td>[CEA-766-C]</td>
<td>April 2008</td>
<td>U.S. and Canadian Rating Region Table (RRT) and Content Advisory Descriptor for Transport of Content Advisory Information Using ATSC Program and System Information Protocol (PSIP).</td>
<td></td>
<td>NA</td>
</tr>
<tr>
<td>[DVB-GEM 1.0.2]</td>
<td></td>
<td>ETSI TS 102 819 V1.3.1 (2005-10) DVB Globally Executable MHP version 1.0.2,(GEM 1.0.2), ETSI TS 102 819 V1.3.1 (2005-10)</td>
<td></td>
<td>NA</td>
</tr>
<tr>
<td>[DVB-MHP 1.0.3]</td>
<td></td>
<td>ETSI TS 101 812 V1.3.2 (2006-08) DVB Multimedia Home Platform 1.0.3, DVB-MHP 1.0.3, ETSI TS 101 812 V1.3.2 (2006-08)</td>
<td></td>
<td>NA</td>
</tr>
<tr>
<td>References</td>
<td>Edition</td>
<td>Description</td>
<td>See Notes Section</td>
<td>MHP Index</td>
</tr>
<tr>
<td>----------------</td>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>[HAVi] 1.1</td>
<td></td>
<td>HAVi Level 2 User Interface specification, HAVi 1.1 <a href="http://www.havi.org/">http://www.havi.org/</a></td>
<td></td>
<td>NA</td>
</tr>
<tr>
<td>[Java RMI]</td>
<td>Rev 1.8, Java 2 SDK Std Ed, v1.4</td>
<td>Java Remote Method Invocation Specification, Java RMI, Rev 1.8, Java 2 SDK Std Ed, v1.4.</td>
<td></td>
<td>[33]</td>
</tr>
<tr>
<td>[Java TV]</td>
<td>Version 1.1.1</td>
<td>Java TV API Specification, Java TV, Version 1.1.1.</td>
<td></td>
<td>[52]</td>
</tr>
<tr>
<td>[JFIF]</td>
<td></td>
<td>JPEG File Interchange Format, Eric Hamilton, C-Cube Microsystems, JFIF.</td>
<td>[5]</td>
<td>[36]</td>
</tr>
<tr>
<td>[JSSE]</td>
<td>SECOP Version 1.0.1</td>
<td>Security (JSSE) package specification v1.0.1.</td>
<td></td>
<td>[61]</td>
</tr>
</tbody>
</table>
2.2.2 Informative References

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: Annex C and are extensions of [DVB-MHP 1.0.3] Section: Annex C.

The [DVB-MHP 1.0.3] reference Table should be replaced with the following:

<table>
<thead>
<tr>
<th>Reference</th>
<th>Edition</th>
<th>Description</th>
<th>See Notes Section 2.2.1.1.1</th>
<th>MHP Index</th>
</tr>
</thead>
</table>

Table 2–3 - OCAP Informative References
<table>
<thead>
<tr>
<th>Reference</th>
<th>Edition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CEA-775-C]</td>
<td>Sept 2008</td>
<td>DTV 1394 Interface Specification, CEA-775-C (ANSI)</td>
</tr>
<tr>
<td>[JVM Tech]</td>
<td></td>
<td>JVM Technology <a href="http://java.sun.com/j2se/1.4.2/docs/guide/misc/threadPrimitiveDeprecation.html">http://java.sun.com/j2se/1.4.2/docs/guide/misc/threadPrimitiveDeprecation.html</a></td>
</tr>
</tbody>
</table>

*See Notes Section 2.2.1.1.1*
3 Glossary

This section contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 3 Definitions and abbreviations and are extensions of [DVB-MHP 1.0.3] 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 1.0.2] Section. This section corresponds to [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2] as follows:

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>MHP Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 Glossary</td>
<td>3 Definitions and abbreviations</td>
<td>Extension</td>
<td>3 Definitions and abbreviations</td>
<td>Extension</td>
</tr>
<tr>
<td>3.1 DVB-GEM and DVB-MHP Specification Correspondence</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>3.2 OCAP Specific Requirements</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>3.2.1 Definitions</td>
<td>3.1 Definitions</td>
<td>Extension</td>
<td>3.1 Definitions</td>
<td>Extension</td>
</tr>
</tbody>
</table>

3.2 OCAP Specific Requirements

This subsection is an OCAP specific section that does not correspond to any [DVB-GEM 1.0.2] Section.

3.2.1 Definitions

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 3.1 Definitions and are extensions of [DVB-MHP 1.0.3] 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 that 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).

BNF

See Backus Naur Form (BNF).

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.

Bound application

Bound applications are those applications which are bound to, or associated with, a particular service made available by the cable operator.

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.
<table>
<thead>
<tr>
<th><strong>Broadcast File System (BFS)</strong></th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CA</strong></td>
<td>See Conditional access (CA) and encryption.</td>
</tr>
<tr>
<td><strong>Cable environment</strong></td>
<td>The technologies, rules, and policies provided by the OCAP specification and the monitor application.</td>
</tr>
<tr>
<td><strong>Cable operator</strong></td>
<td>See Multiple Systems Operator (MSO).</td>
</tr>
<tr>
<td><strong>CableCARD</strong></td>
<td>A CableCARD device, also referred to as &quot;Point of Deployment&quot; (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.</td>
</tr>
<tr>
<td><strong>Carousel</strong></td>
<td>See Object carousel.</td>
</tr>
<tr>
<td><strong>CATV</strong></td>
<td>An abbreviation for &quot;Cable TV&quot;.</td>
</tr>
<tr>
<td><strong>CCI</strong></td>
<td>See Copy Control Information (CCI).</td>
</tr>
<tr>
<td><strong>Certificate Revocation List (CRL)</strong></td>
<td>A list of revoked certificates published by each certificate authority.</td>
</tr>
<tr>
<td><strong>Conditional access (CA) and encryption</strong></td>
<td>Conditional access and encryption is a system that provides selective access to programming to individual customers in exchange for payment.</td>
</tr>
<tr>
<td><strong>Content</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Content protection/copy protection (CP)</strong></td>
<td>Content protection is a mechanism to protect the unauthorized copying of video and audio programming.</td>
</tr>
<tr>
<td><strong>Coordinated Universal Time (UTC)</strong></td>
<td>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).</td>
</tr>
<tr>
<td><strong>Copy Control Information (CCI)</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Corrigenda</strong></td>
<td>Errata.</td>
</tr>
<tr>
<td><strong>CP</strong></td>
<td>See Content protection/copy protection (CP).</td>
</tr>
<tr>
<td><strong>CRC</strong></td>
<td>See Cyclic Redundancy Check (CRC).</td>
</tr>
<tr>
<td><strong>CRL</strong></td>
<td>See Certificate Revocation List (CRL).</td>
</tr>
<tr>
<td><strong>Cross-environment applications</strong></td>
<td>Applications able to run in multiple environments. The terms &quot;cross-environment application&quot; and &quot;application running in cross-environment mode&quot; are equivalent. Cross environment applications are the union of cross-service applications and utility applications.</td>
</tr>
</tbody>
</table>
Cross-service applications | Those cross-environment applications that 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 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 Multi-Function Host 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 generalised opt-in by the end-user is not a direct request.
DOCSIS® | See Data-Over-Cable Service Interface Specifications.
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.

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

In the broadcast case, a connection corresponds to a DVB-service. Broadcast 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-MHP 1.0.3]. For the OCAP 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-MHP 1.0.3]. For the OCAP 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-MHP 1.0.3] are not directly supported by OCAP 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.
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Elementary Stream (ES)</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Emergency Alert System (EAS)</strong></td>
<td>The US Federal system for alerting the public to emergencies. EAS is a digital upgrade to the old Emergency Broadcasting System.</td>
</tr>
<tr>
<td><strong>EN</strong></td>
<td>European Norms. Prefix for certain ETSI documents.</td>
</tr>
<tr>
<td><strong>Environment</strong></td>
<td>A set of technologies, rules, and policies that support the provision of one or more services to the end-user.</td>
</tr>
<tr>
<td><strong>Environment States</strong></td>
<td>Each environment is always in one of four states: Selected, Background, Presenting, or Inactive (see Section 10.2.2.4 for details).</td>
</tr>
<tr>
<td><strong>EPG</strong></td>
<td>See Electronic Program Guide (EPG).</td>
</tr>
<tr>
<td><strong>ES</strong></td>
<td>See Elementary Stream (ES).</td>
</tr>
<tr>
<td><strong>ETS</strong></td>
<td>European Telecommunications Standard. Prefix for certain ETSI documents.</td>
</tr>
<tr>
<td><strong>ETSI</strong></td>
<td>European Telecommunications Standard Institute.</td>
</tr>
<tr>
<td><strong>Events</strong></td>
<td>Events are asynchronous communication between applications and the OpenCable system on which they are being executed. They provide communication between solution elements. 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.</td>
</tr>
<tr>
<td><strong>Execution Engine (EE)</strong></td>
<td>The Execution Engine is a platform-independent interface that permits programmatic content as part of the OpenCable Application Platform</td>
</tr>
<tr>
<td><strong>FCC</strong></td>
<td>See Federal Communications Commission (FCC).</td>
</tr>
<tr>
<td><strong>Federal Communications Commission (FCC)</strong></td>
<td>A U.S. Federal agency responsible for establishing policies to govern interstate and international communications.</td>
</tr>
<tr>
<td><strong>Function</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>GEM</strong></td>
<td>See Globally Executable MHP.</td>
</tr>
<tr>
<td><strong>Global System for Mobile Communications (GSM)</strong></td>
<td>An international standard, developed in Europe, for digital mobile communications.</td>
</tr>
<tr>
<td><strong>Globally Executable MHP</strong></td>
<td>A terminal specification based on MHP that enables applications to interoperate across OCAP, MHP and other GEM based platforms.</td>
</tr>
<tr>
<td><strong>GSM</strong></td>
<td>See Global System for Mobile Communications (GSM).</td>
</tr>
<tr>
<td><strong>HAVi</strong></td>
<td>See Home Audio/Video interoperability (HAVi) architecture.</td>
</tr>
<tr>
<td><strong>HDTV</strong></td>
<td>See High Definition Television (HDTV).</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Head-end</td>
<td>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.</td>
</tr>
<tr>
<td>High Definition Television (HDTV)</td>
<td>Television that substantially exceeds NTSC, PAL or SECAM in resolution and quality.</td>
</tr>
<tr>
<td>Home Audio/Video interoperability (HAVi) architecture</td>
<td>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.</td>
</tr>
<tr>
<td>Home environment</td>
<td>The environment to which an application belongs.</td>
</tr>
<tr>
<td>Host device</td>
<td>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.</td>
</tr>
<tr>
<td>Host device manufacturer application</td>
<td>OCAP application deployed to an 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. Note that host device manufacturer applications may be composed of both OCAP-J and native elements.</td>
</tr>
<tr>
<td>HTTP</td>
<td>See Hypertext Transport Protocol (HTTP).</td>
</tr>
<tr>
<td>Hypertext Transport Protocol (HTTP)</td>
<td>HTTP is the transport layer for HTML documents over the Internet Protocol (IP).</td>
</tr>
<tr>
<td>IETF</td>
<td>See Internet Engineering Task Force (IETF).</td>
</tr>
<tr>
<td>Interactive Television (ITV)</td>
<td>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.</td>
</tr>
<tr>
<td>International Organization for Standardization (ISO)</td>
<td>An international standards body.</td>
</tr>
<tr>
<td>International Telecommunication Union (ITU)</td>
<td>An international organization within which governments and the private sector coordinate global telecom networks and services.</td>
</tr>
<tr>
<td>Internet Engineering Task Force (IETF)</td>
<td>A cooperative consortium that standardizes internet protocols, naming and other communications issues.</td>
</tr>
<tr>
<td>Internet Protocol (IP)</td>
<td>A layer 3 communications protocol commonly used in the internet. It is defined by RFC791.</td>
</tr>
<tr>
<td>IP</td>
<td>See Internet Protocol (IP).</td>
</tr>
<tr>
<td>ITU</td>
<td>See International Telecommunication Union (ITU).</td>
</tr>
<tr>
<td>ITV</td>
<td>See Interactive Television (ITV).</td>
</tr>
<tr>
<td>Java Development Kit (JDK)</td>
<td>A set of resources, including software and documents, provided by SUN Microsystems to enable developers to program in the Java language.</td>
</tr>
<tr>
<td><strong>Java Media Framework (JMF)</strong></td>
<td>A Java package providing functionality primarily for data streaming, provided as part of the Java TV specification.</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Java Secure Socket Extension (JSSE)</strong></td>
<td>A Java package providing functionality for secure network communications.</td>
</tr>
<tr>
<td><strong>Java™ API</strong></td>
<td>The Java API is a standard interface for use by platform-independent application software. It is expressed in the Java language.</td>
</tr>
<tr>
<td><strong>JDK</strong></td>
<td>See Java Development Kit (JDK).</td>
</tr>
<tr>
<td><strong>JFIF</strong></td>
<td>See JPEG File Interchange Format ([JFIF]).</td>
</tr>
<tr>
<td><strong>JMIF</strong></td>
<td>See Java Media Framework (JMF).</td>
</tr>
<tr>
<td><strong>JPEG File Interchange Format ([JFIF])</strong></td>
<td>A platform-agnostic JPEG file format.</td>
</tr>
<tr>
<td><strong>JSSE</strong></td>
<td>A java package containing the CDC Security (Java Secure Socket Extension - JSSE) package interfaces and modules.</td>
</tr>
<tr>
<td><strong>Key handling epoch</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Lifetime of an application</strong></td>
<td>The lifetime of an application characterizes the time from which the application is loaded to the time the application is destroyed.</td>
</tr>
<tr>
<td><strong>LMDS</strong></td>
<td>See Local Multipoint Distribution System (LMDS).</td>
</tr>
<tr>
<td><strong>Local Multipoint Distribution System (LMDS)</strong></td>
<td>A fixed wireless technology that is one solution for bringing high-bandwidth services to homes and offices within the &quot;last mile&quot; of connectivity.</td>
</tr>
<tr>
<td><strong>Locator</strong></td>
<td>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 environment.</td>
</tr>
<tr>
<td><strong>MA</strong></td>
<td>See Monitor Application (MA).</td>
</tr>
<tr>
<td><strong>MAC</strong></td>
<td>See Media Access Control (MAC).</td>
</tr>
<tr>
<td><strong>Mandatory Ordinary Keycodes</strong></td>
<td>The Mandatory Ordinary keycodes are specific key codes that can't be filtered by OCAP event filtering.</td>
</tr>
<tr>
<td><strong>Man-Machine Interface (MMI)</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Manufacturer environment</strong></td>
<td>One particular example of a non-cable environment that is provided by the manufacturer of the CE device.</td>
</tr>
<tr>
<td><strong>Media Access Control (MAC)</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>
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.

MIME

See Multipurpose Internet Mail Extensions (MIME).

MMDS

See Multipoint Microwave Distribution System (MMDS).

MMI

See Man-Machine Interface (MMI).

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.

MPEG

A prefix for a set of file formats pertaining to video and audio data. See Moving Picture Expert Group (MPEG).

MSO

See Multiple Systems Operator (MSO).

Multi-Function Host

An OpenCable Host device that implements both cable and non-cable environments.

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 Microwave 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 device. Typically, it is written in C, C++, or assembly language and may be supplied with the OCAP implementation or downloaded over the cable.

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 environment

A set of technologies rules, and policies other than that provided by the OCAP specification and the monitor application.

Non-OCAP application

Applications that are not listed in the OCAP Applications Database, do not observe the OCAP resource management mechanisms, are supplied with the OCAP implementation by the manufacturer of the OCAP host device or other party, and are associated with a manufacturer application environment.

Non-shared resource

Resources that 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 API

OCAP API refers to one of the Java APIs standardized as part of OCAP. For the OCAP implementation, the OCAP APIs include the DVB-J APIs that have been modified and/or extended by this specification.

OCAP implementation

The OCAP implementation is the actual software that provides support for the defined OCAP on a host receiver.

OCAP application

Applications that are listed in the OCAP Applications Database, observe the OCAP resource management mechanisms for both shared and OCAP resources, and are associated with the cable application environment.

OCAP-J

OCAP-J is an application type.

OCAP-J proxy application

An OCAP 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.
**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 Bundle**
Defines a set of specifications required to build a specific version of an OpenCable device. See [OC-BUNDLE].

**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.

**ocap.profile**
Reserved for the profile of the Host device. There are no active profiles at this time; values in this field should be ignored.

**OpenCable terminal**
See OpenCable device.

**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.

**OS**
See Operating System (OS).

**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)**
a MPEG stream is composed of one or more elementary streams (ES), each containing audio, video, or data. ES's can be grouped into Program Streams, which are formed by breaking ES's into chunks, the PES's, and interleaving them.

**Pauseable 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 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).

**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.
<table>
<thead>
<tr>
<th><strong>Request For Comment (RFC)</strong></th>
<th>A document and a process for establishing standards. IETF standards are documented as RFCs.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Resident application</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Return channel</strong></td>
<td>A return channel refers to the communications mechanism that provides connection between OpenCable and a remote server.</td>
</tr>
<tr>
<td><strong>RFC</strong></td>
<td>See Request For Comment (RFC).</td>
</tr>
<tr>
<td><strong>RMI</strong></td>
<td>See Remote Method Invocation (RMI).</td>
</tr>
<tr>
<td><strong>Sandbox</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Satellite Master Antenna TV (SMATV)</strong></td>
<td>RF distribution of satellite and antenna signals.</td>
</tr>
<tr>
<td><strong>SCTE</strong></td>
<td>See Society of Cable Telecommunications Engineers (SCTE).</td>
</tr>
<tr>
<td><strong>Secure Sockets Layer (SSL)</strong></td>
<td>A public key encryption based protocol for secure communications between client and server.</td>
</tr>
<tr>
<td><strong>Selected environment</strong></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>NOTE: See Section 10.2.2.4.1 for details on when and how environments become selected.</td>
</tr>
<tr>
<td><strong>Service</strong></td>
<td>A service is a sequence of programs under the control of a broadcaster which can be broadcast as part of a schedule.</td>
</tr>
<tr>
<td><strong>Service application</strong></td>
<td>An application is service-bound if, and only if, it is associated with one or more broadcast services.</td>
</tr>
<tr>
<td><strong>Service Information (SI)</strong></td>
<td>That information that describes the broadcast services available on the network.</td>
</tr>
<tr>
<td><strong>Shared resource</strong></td>
<td>Resources that 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.</td>
</tr>
<tr>
<td><strong>SI</strong></td>
<td>See Service Information (SI).</td>
</tr>
<tr>
<td><strong>SMATV</strong></td>
<td>See Satellite Master Antenna TV (SMATV).</td>
</tr>
<tr>
<td><strong>Society of Cable Telecommunications Engineers (SCTE)</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>SSL</strong></td>
<td>See Secure Sockets Layer (SSL).</td>
</tr>
<tr>
<td><strong>TLS</strong></td>
<td>See Transport Layer Security (TLS).</td>
</tr>
</tbody>
</table>
### 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.

### Transport Layer Security (TLS)

An Internet security protocol based on SSL.

### 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 Multi-Function Host, e.g., volume control (non-OCAP applications).

### VBI

See Vertical Blanking Interval (VBI).

### VCT

See Virtual Channel Table (VCT).

### 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.

### 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 1.0.2] Section: 3 Definitions and abbreviations and are extensions of [DVB-MHP 1.0.3] 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 1.0.2] Section. Section 4 (this section) corresponds to [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2] as follows:

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>MHP Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Acronyms</td>
<td>3 Definitions and abbreviations</td>
<td>Extension</td>
<td>3 Definitions and abbreviations</td>
<td>Extension</td>
</tr>
<tr>
<td>4.1 DVB-GEM and DVB-MHP Specification Correspondence</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>4.2 OCAP Specific Requirements</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>4.2.1 Acronyms</td>
<td>3.2 Abbreviations</td>
<td>Extension</td>
<td>3.2 Abbreviations</td>
<td>Extension</td>
</tr>
</tbody>
</table>

4.2 OCAP Specific Requirements

This subsection is an OCAP specific section that does not correspond to any [DVB-GEM 1.0.2] Section.

4.2.1 Acronyms

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 3.2 Abbreviations and are extensions of [DVB-MHP 1.0.3] 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
<table>
<thead>
<tr>
<th>CHILA</th>
<th>CableCARD Host Interface License Agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORBA</td>
<td>Common Object Request Broker Architecture</td>
</tr>
<tr>
<td>CP</td>
<td>Content Protection</td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>CRL</td>
<td>Certificate Revocation List</td>
</tr>
<tr>
<td>DAVIC</td>
<td>Digital Audio Visual Council</td>
</tr>
<tr>
<td>DECT</td>
<td>Digital Enhanced Cordless Telecommunications</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Configuration Protocol</td>
</tr>
<tr>
<td>DII</td>
<td>DownloadInfoIndication</td>
</tr>
<tr>
<td>DOCSIS</td>
<td>Data-Over-Cable Service Interface Specifications</td>
</tr>
<tr>
<td>DSM-CC</td>
<td>Digital Storage Media—Command and Control. Part 6 of the MPEG-2 standard</td>
</tr>
<tr>
<td>DSM-CC-OC</td>
<td>Digital Storage Media—Command and Control Object Carousel</td>
</tr>
<tr>
<td>DSM-CC-UU</td>
<td>Digital Storage Media—Command and Control User to User</td>
</tr>
<tr>
<td>DTD</td>
<td>Document Type Definition</td>
</tr>
<tr>
<td>DVB</td>
<td>Digital Video Broadcasting</td>
</tr>
<tr>
<td>DVS</td>
<td>Digital Video Subcommittee</td>
</tr>
<tr>
<td>DTVCC</td>
<td>Digital Television Closed Captioning</td>
</tr>
<tr>
<td>EAS</td>
<td>Emergency Alert System</td>
</tr>
<tr>
<td>ECM</td>
<td>Entitlement Control Message</td>
</tr>
<tr>
<td>EE</td>
<td>Execution Engine</td>
</tr>
<tr>
<td>EIA</td>
<td>Electronic Industry Alliance</td>
</tr>
<tr>
<td>EN</td>
<td>European Norms</td>
</tr>
<tr>
<td>EPG</td>
<td>Electronic Program Guide</td>
</tr>
<tr>
<td>ES</td>
<td>Elementary Stream</td>
</tr>
<tr>
<td>ETS</td>
<td>European Telecommunications Standard</td>
</tr>
<tr>
<td>ETSI</td>
<td>European Telecommunications Standard Institute</td>
</tr>
<tr>
<td>ExCCI</td>
<td>Extended Copy Control Information</td>
</tr>
<tr>
<td>FCC</td>
<td>Federal Communications Commission</td>
</tr>
<tr>
<td>GEM</td>
<td>Globally Executable MHP</td>
</tr>
<tr>
<td>GSM</td>
<td>Global System for Mobile Communications</td>
</tr>
</tbody>
</table>
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
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 Microwave 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
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OOB-RDC</td>
<td>Out-Of-Band Reverse-Data-Channel</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PCR</td>
<td>Program Clock Reference</td>
</tr>
<tr>
<td>PES</td>
<td>Packetized Elementary Stream</td>
</tr>
<tr>
<td>PID</td>
<td>Packet Identification number</td>
</tr>
<tr>
<td>PMT</td>
<td>Program Map Table</td>
</tr>
<tr>
<td>POD</td>
<td>Point of Deployment module</td>
</tr>
<tr>
<td>PPV</td>
<td>Pay-per-view</td>
</tr>
<tr>
<td>PSIP</td>
<td>Program and System Information Protocol</td>
</tr>
<tr>
<td>PSTN</td>
<td>Public Switched Telephone Network</td>
</tr>
<tr>
<td>PVR</td>
<td>Personal Video Recorder</td>
</tr>
<tr>
<td>RCMM</td>
<td>Root Certificate Management Messages</td>
</tr>
<tr>
<td>RMI</td>
<td>Remote Method Invocation</td>
</tr>
<tr>
<td>SCTE</td>
<td>Society of Cable Telecommunications Engineers</td>
</tr>
<tr>
<td>SDP</td>
<td>Session Description Protocol</td>
</tr>
<tr>
<td>SI</td>
<td>Service Information</td>
</tr>
<tr>
<td>SMATV</td>
<td>Satellite Master Antenna TV</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer</td>
</tr>
<tr>
<td>STC</td>
<td>System Time Clock</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>URI</td>
<td>Universal Resource Identifier</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>UTC</td>
<td>Coordinated Universal Time</td>
</tr>
<tr>
<td>VBI</td>
<td>Vertical Blanking Interval</td>
</tr>
<tr>
<td>VCT</td>
<td>Virtual Channel Table</td>
</tr>
<tr>
<td>VOD</td>
<td>Video-On-Demand</td>
</tr>
<tr>
<td>XAIT</td>
<td>Extended Application Information Table</td>
</tr>
</tbody>
</table>
5 CONVENTIONS

This section contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 4 General considerations and conventions.

The following convention is used in this specification:

Courier New font is used to indicate code examples, names of properties, and other information that SHALL be entered exactly as-is: code example font.

5.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

<table>
<thead>
<tr>
<th>Table 5–1 - Correlation between OCAP and [DVB-GEM 1.0.2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCAP</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>5 Conventions</td>
</tr>
<tr>
<td>5.1 DVB-GEM and DVB-MHP Specification Correspondence</td>
</tr>
<tr>
<td>5.2 OCAP Specific Requirements</td>
</tr>
<tr>
<td>5.2.1 Specification Language</td>
</tr>
<tr>
<td>5.2.2 Organization of the OCAP Specification</td>
</tr>
<tr>
<td>No Corresponding Section</td>
</tr>
<tr>
<td>No Corresponding Section</td>
</tr>
</tbody>
</table>

5.2 OCAP Specific Requirements

This subsection is an OCAP specific section that does not correspond to any [DVB-GEM 1.0.2] Section.

5.2.1 Specification Language

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] 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 1.0.2] Specification as its base. Where applicable, OCAP references the corresponding section within [DVB-GEM 1.0.2]. However, this document does not follow the same organization as the [DVB-GEM 1.0.2] 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:

- This section is compliant with DVB-GEM 1.0.2 Section: <section name>.
- This section contains requirements that are extensions to DVB-GEM 1.0.2 Section: <section name>.
- This section is compliant with DVB-GEM 1.0.2 Section: <section name> and an <extension> of DVB-MHP 1.0.3 Section: <section name>.
- This section contains requirements that are extensions to DVB-GEM 1.0.2 Section: <section name> and an <extension> of DVB-MHP 1.0.3 Section: <section name>.
- This section is an OCAP-specific requirement not corresponding to any DVB-GEM 1.0.2 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:

- Informative explanations of a specification may refer to a DVB-MHP implementation. These will not be included in conformance Tables.
- Extensions or deviations of a DVB-MHP implementation of DVB-GEM requirements (which OCAP conforms to), may be defined in OCAP. In this case the appropriate DVB-MHP sections will be specifically addressed in conformance statements like those used for DVB-GEM sections.

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 is 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 1.03 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:

<table>
<thead>
<tr>
<th>OCAP Section</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>Compliance to DVB-GEM</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>Compliance to DVB-MHP</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Complete Compliance&quot;, or &quot;Extension&quot;, or &quot;Deviation&quot;</td>
<td>&quot;Complete Compliance&quot;, or &quot;Extension&quot;, or &quot;Deviation&quot;</td>
<td>&quot;Complete Compliance&quot;, or &quot;Extension&quot;, or &quot;Deviation&quot;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 1.0.2] Section.

6.1 GEM Compliance

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

OCAP terminals SHALL comply in full with [DVB-GEM 1.0.2]. For avoidance of doubt, in the event of a conflict between [DVB-GEM 1.0.2] and this specification, the normative guarantees of [DVB-GEM 1.0.2] SHALL take precedence except as detailed in Section 6.1.1. Due to the obsolescence of PersonalJava, OCAP extends [DVB-GEM 1.0.2] in general to comply with [JSR 217]. OCAP terminals SHALL include a complete PBP 1.1 implementation as defined in [JSR 217].

Because OCAP is a GEM Terminal Specification (as defined in [DVB-GEM 1.0.2]), the following terminology applies:

- An OCAP 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 1.0.2] is required.

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 GEM and MHP Correspondence</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
<tr>
<td>6.1 GEM Compliance</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
<tr>
<td>6.2 DVB-GEM and DVB-MHP Full Compliance</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
<tr>
<td>(informative)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.1.1 GEM Errata

Following are errata to [DVB-GEM 1.0.2]. 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 1.0.2] clause G is considered to read:

MHP [1], 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.

### 6.1.1.3 Functional Equivalents

[DVB-GEM 1.0.2] table 8 errata makes application authentication a functional equivalent. To comply with the errata the row in that table with name "Application Authentication" SHALL be read as follows:

**Table 6–1 - Application Authentication Functional Equivalent**

<table>
<thead>
<tr>
<th>Name</th>
<th>GEM Clause(s)</th>
<th>MHP Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NOTE 1: See also text in clause 12.6.2.6, &quot;Credentials&quot; and text in clause 12.9, &quot;Certificate management&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOTE 2: See also text in clause 12.9.2, &quot;Root certificate management&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M M M M</td>
<td>E.P. I.P. E.P. I.P.</td>
</tr>
</tbody>
</table>

### 6.1.1.4 javax.tv.util.TVTimer Clarifications

The object returned by the TVTimer.scheduleTimerSpec(TVTimerSpec) method SHALL be the same object instance as was passed in as an argument to the method.

NOTE: [DVB-MHP 1.0.3] clause A.5.2.2, "deschedule(TVTimerSpec)", requires that the argument be an instance returned from the method TVTimer.scheduleTimerSpec(TVTimerSpec). This is not a requirement of [Java TV], but GEM terminal implementations are required to obey this MHP-defined behavior.

NOTE: [Java TV] requires that the argument to deschedule(TVTimerSpec) be the instance passed as an argument to the method TVTimer.scheduleTimerSpec(TVTimerSpec). Implementations of GEM terminal specifications based on [Java TV] are required to obey this JavaTV-defined behavior. This may be implemented in a way consistent with the MHP requirements by implementing TVTimer.scheduleTimerSpec(TVTimerSpec) such that it always returns the instance passed in as an argument. Note that the specification of TVTimer.scheduleTimerSpec(TVTimerSpec) in [Java TV] allows the implementation to modify the instance passed in as argument, e.g., to account for timer granularity.

The specification for the TVTimer.scheduleTimerSpec method contains the statement, "The actual listener notification may happen asynchronously." This statement SHALL be read as, "The actual listener notification SHALL occur on a thread separate from the thread calling this method."

### 6.1.2 GEM References

As a consequence of PBP 1.1 compliance as stated in Section 6.1, Java specification references in [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3] have been updated in the following sections.
6.1.2.1 DVB-J Class Files

[DVB-MHP 1.0.3] section 6.2.5.1 contains the statement, "Java bytecode for each Java class is carried as the content bytes of the BIOP::FileMessage corresponding exactly to the contents of a "class" file as specified in Java VM [34]." This statement SHALL be read as "Java bytecode for each Java class is carried as the content bytes of the BIOP::FileMessage corresponding exactly to the contents of a "class" file as specified in the Java virtual machine specification volume 2, as referenced by Personal Basis Profile (PBP) 1.1, JSR 217"; see [JSR 217].

6.1.2.2 Monomedia format for text

[DVB-MHP 1.0.3] section 7.1.5 contains the statement, "Java modified UTF-8 as defined in Java Language Spec [32] section 22.2.14 "writeUTF" is the coding of text in MHP." This statement SHALL be read as, "Java modified UTF-8 as defined in the specification of the method java.io.DataOutput.writeUTF."  

6.1.2.3 The Virtual Machine

[DVB-MHP 1.0.3] section 11.1 contains the statements:

"The DVB-J virtual machine is defined in Java VM [34], as amended by JVM Errata [68] plus the Inner Classes specification in Inner Classes [69]."

"The Java Virtual Machine shall support Java class files whose version number is in the range 45.3 through 45.65535."

These statements SHALL be read as, "The Java platform is defined in Personal Basis Profile (PBP) 1.1, JSR 217"; see [JSR 217].

6.1.2.4 Basic Considerations

[DVB-MHP 1.0.3] section 11.2.1 contains the statement, "When the application manager terminates the entity that represents the virtual machine in which the application is run, a conformant implementation is permitted to not run application finalizers, as spelled out in section 2.17.9 of the Java VM [34]." This statement SHALL be read as, "When the application manager terminates the entity that represents the virtual machine in which the application is run, a conformant implementation is permitted to not run application finalizers, as spelled out in section 2.17.9 of the Java virtual machine specification volume 2, as referenced by Personal Basis Profile (PBP) 1.1, JSR 217"; see [JSR 217].

6.1.2.5 Unloading

[DVB-MHP 1.0.3] section 11.2.4 contains the statement, "Class unloading as defined by section 12.8 of Java Language Spec [32] and section 2.16.8 of Java VM [34] will be supported." The statement SHALL be read as "Class unloading as required by Personal Basis Profile (PBP) 1.1, JSR 217 SHALL be supported"; see [JSR 217].

6.1.2.6 Java Platform APIs

[DVB-MHP 1.0.3] sections 11.3.1 and 11.3.1.1 contain references to "JAE 1.1.8 API [31]", "PersonalJAE [36]", and "JDK 1.2.2 [78]". Each of these references SHALL be read as "Personal Basis Profile (PBP) 1.1, JSR 217"; see [JSR 217].

6.1.2.7 java.util

[DVB-MHP 1.0.3] section 11.3.1.3 contains a reference to "JDK 1.2.2 [78]". This reference SHALL be read as "Personal Basis Profile (PBP) 1.1, JSR 217"; see [JSR 217].
6.1.2.8 **org.dvb.lang**

[DVB-MHP 1.0.3] section 11.3.2.1 contains a reference to "PersonalJAE [36]". This reference SHALL be read as "Personal Basis Profile (PBP) 1.1, JSR 217"; see [JSR 217].

6.1.2.9 **Graphics Reference Model**

[DVB-MHP 1.0.3] sections 13.3.4.1, 13.3.6, and 13.6.1.2.2 contain a references to "JAE 1.1.8 API [31]". These references SHALL be read as "Personal Basis Profile (PBP) 1.1, JSR 217"; see [JSR 217].

6.1.2.10 **Public and Protected final static primitive fields from standard Java packages**

[DVB-MHP 1.0.3] section 16.2.2 contains references to "JAE 1.1.8 const [72]" and "JAE 1.2.2 const [73]". Each of these references SHALL be read as "Personal Basis Profile (PBP) 1.1, JSR 217"; see [JSR 217].

6.1.2.11 **External references; errata, clarifications and exemptions**

[DVB-MHP 1.0.3] sections A.1, A.2, A.4, and A.9 contain references to "JAE 1.1.8 API [31]", "Java Language Spec [32]", "Java VM [34]", and "JAE 1.1.8 const [72]". These references SHALL be read as "Personal Basis Profile (PBP) 1.1, JSR 217"; see [JSR 217].

**NOTE:** The errata in [DVB-MHP 1.0.3] sections A.1, A.2, A.4, and A.9 are effectively included in the definition of the Personal Basis Profile (PBP) 1.1. As such, these clauses can be effectively disregarded without affecting DVB-GEM compliance.

6.1.2.12 **org.dvb.application.CurrentServiceFilter**

The OCAP Javadoc class description for org.dvb.application.CurrentServiceFilter reads:

A CurrentServiceFilter is used to indicate that only applications that are signalled as part of the current service shall be returned by the getAppsAttributes and getAppIDs methods of AppsDatabase. Externally authorized applications in the AIT are not considered to be signalled as part of the current service for this filter. Subclasses of CurrentServiceFilter can override the accept method so as to implement their own filter criteria on the AppIDs values.

To match the verbiage specified in [DVB-MHP 1.0.3], the OCAP Javadoc class description for org.dvb.application.CurrentServiceFilter SHALL read:

A CurrentServiceFilter is used to indicate that only applications that are signalled as part of the current service shall be returned by the getAppAttributes and getAppIDs methods of AppsDatabase. Externally authorized applications in the AIT are not considered to be signalled as part of the current service for this filter. If an application signaled as part of the current service has an application instance in the destroyed state, then information on that application instance shall not be retrieved. Instead, what shall be retrieved is information on another application instance which would normally be in the not loaded state. Subclasses of CurrentServiceFilter can override the accept method so as to implement their own filter criteria on the AppIDs values.

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

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

As a consequence of Section 6.1, OCAP is fully compliant with the following [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3] sections. This section maps out the sections of the OCAP specification that are not specifically called out in
the compliance Tables in the individual sections of the OCAP to the [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3] 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>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3]</th>
<th>MHP Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annex A (deleted)</td>
<td>Annex A (normative): External references; errata, clarifications and exemptions</td>
<td>Compliance</td>
<td>Annex A (normative): External references; errata, clarifications and exemptions</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex F (informative): Authoring and implementation guidelines</td>
<td>Compliance</td>
<td>Annex F (informative): Authoring and Implementation Guidelines</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex K (normative): DVB-J persistent storage API</td>
<td>Compliance</td>
<td>Annex K (normative): DVB-J persistent storage API</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex L (normative): User settings and preferences API</td>
<td>Compliance</td>
<td>Annex L (normative): User Settings and Preferences API</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex O (normative): Integration of the JavaTV SI API</td>
<td>Compliance</td>
<td>Annex O (normative): Integration of the JavaTV SI API and DVB SI</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex S (normative): Application listing and launching</td>
<td>Compliance</td>
<td>Annex S (normative): Application Listing and Launching</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex U (normative): Extended graphics APIs</td>
<td>Compliance</td>
<td>Annex U (normative): Extended graphics APIs</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex V: Void</td>
<td>Compliance</td>
<td>Annex V: Void</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex W (informative): DVB-J examples</td>
<td>Compliance</td>
<td>Annex W (normative): DVB-J examples</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex X (normative): Test support</td>
<td>Compliance</td>
<td>Annex X (normative): Test support</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex Y (normative): Inter-application and Inter-Xlet communication API</td>
<td>Compliance</td>
<td>Annex Y (normative): Inter-application communication API</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex Z (informative): Services, service contexts and applications in an MHP environment</td>
<td>Compliance</td>
<td>Annex Z (informative): Services, Service Contexts and Applications in an MHP Environment</td>
<td>Compliance</td>
</tr>
</tbody>
</table>

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 application should be substituted for the phrase GEM application or MHP application. Similarly, OCAP environment should be substituted for GEM environment or MHP environment and the term OCAP 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 replaces GEM and MHP; for example, in the following terms and phrases:

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

These sections in [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3] are not reprinted in OCAP. Corresponding OCAP sections are not written.
7 BASIC ARCHITECTURE

This section is compliant with [DVB-GEM 1.0.2] Section: 5 Basic architecture and are extensions of [DVB-MHP 1.0.3] Section: 5 Basic Architecture.

OCAP 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 API provides an abstraction layer to the set-top box's operating system that manages the underlying hardware's resources.

This section describes how the OCAP 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 1.0.2] Section. Section 7 (this section) of OCAP corresponds to Section 5 of [DVB-MHP 1.0.3] as follows ([DVB-GEM 1.0.2] does not define an architecture):

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>MHP Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 Basic Architecture</td>
<td>5 Basic Architecture</td>
<td>Compliance</td>
<td>5 Basic Architecture</td>
<td>Extension</td>
</tr>
<tr>
<td>7.1 DVB-GEM and DVB-MHP Specification Correspondence</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>7.2 OCAP Specific Requirements</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>7.2.1 Deviations from DVB-MHP</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>7.2.1.1 Context</td>
<td>5 Basic architecture</td>
<td>Compliance</td>
<td>5.1 Context.</td>
<td>Extension</td>
</tr>
<tr>
<td>7.2.1.2 Architecture</td>
<td>5 Basic architecture</td>
<td>Compliance</td>
<td>5.2 Architecture</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>5 Basic architecture</td>
<td>Compliance</td>
<td>5.2.1 Resources</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>5 Basic architecture</td>
<td>Compliance</td>
<td>5.2.2 System software</td>
<td>Compliance</td>
</tr>
<tr>
<td>7.2.1.3 Application</td>
<td>5 Basic architecture</td>
<td>Compliance</td>
<td>5.2.3 Application</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>5 Basic architecture</td>
<td>Compliance</td>
<td>5.3 Interfaces Between an MHP Application and the MHP System</td>
<td>Compliance</td>
</tr>
<tr>
<td>7.2.1.4 Plug-ins</td>
<td>5 Basic architecture</td>
<td>Compliance</td>
<td>5.4 Plug-ins</td>
<td>Extension</td>
</tr>
<tr>
<td>7.2.1.5 Informative</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
</tbody>
</table>

7.2 OCAP Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.
7.2.1 Deviations from DVB-MHP

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

7.2.1.1 Context

This subsection is compliant with [DVB-GEM 1.0.2] Section: 5 Basic architecture and is an extension of [DVB-MHP 1.0.3] Section: 5.1 Context.

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

Like MHP, the OCAP software has access to streams of video, audio, data, and other media assets. Because OCAP 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 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 host device. The OCAP 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.

The OCAP 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 is not made available in this version of the specification.

7.2.1.2 Architecture

This subsection is compliant with [DVB-GEM 1.0.2] Section: 5 Basic architecture and is an extension of [DVB-MHP 1.0.3] Section: 5.2 Architecture.

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

The OCAP 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 architecture is described in Section 7.2.1.5.1.
7.2.1.3 Application

This subsection is compliant with [DVB-GEM 1.0.2] Section: 5 Basic architecture and is an extension of [DVB-MHP 1.0.3] Section: 5.2.3 Application.

Figure 7–2 is a more accurate illustration of the OCAP architecture than the one presented in Figure 4, Section 5.2.3 of the [DVB-MHP 1.0.3].

In OCAP, 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 1.0.2] Section: 5 Basic architecture and is an extension of [DVB-MHP 1.0.3] Section: 5.4 Plug-ins.

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

Implementation specific plug-ins are not supported by OCAP.

7.2.1.5 Informative

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

The following information is informative to OCAP. It attempts to clarify issues in the [DVB-MHP 1.0.3] as well as provide additional information to the normative requirements.

7.2.1.5.1 OCAP 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 software architecture is illustrated in Figure 7–2.
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 subsystems specified by this document.
- The OCAP 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 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 Host Device subsystem

OCAP is designed for advanced digital set-top converters. These devices are actually a hybrid analog/digital set-top box. The set-top box supports 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 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 host device will typically be composed 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 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 host device via input devices such as an IR remote control or keyboard.

- Outputs
  In response to viewer input, the OCAP 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 of 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 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 Subsystems

The OCAP 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. 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 Application Programming Interfaces

OCAP 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 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 applications, and arbitration of resource conflicts between OCAP 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 applications that are written to run in any OCAP-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 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 host device by independent content developers.

7.2.1.5.1.4.3 Host device manufacturer application

Host device manufacturer 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 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 Java platform") that abstract the functionality of the television receiver.

7.2.1.5.2.1 The OCAP Java Platform

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

7.2.1.5.2.1.1 Fundamental Java APIs

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

7.2.1.5.2.1.1 Fundamental Java APIs

The OCAP 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 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 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 OCAP.

7.2.1.5.2.1.3 DAVIC APIs

The OCAP 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 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. The DAVIC tune() API SHALL generate the
  NetworkInterfaceTuningEvent and NetworkInterfaceTuningOverEvent for all successful tunes,
  without regard to the prior state of the tuner.

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

• Conditional Access API
  OCAP 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 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 Execution
  Engine.

7.2.1.5.2.1.5 DVB-MHP APIs

The OCAP 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.

• Support Analog Services
  APIs support analog locators; they specifically support:

  It is understood that
  `org.dvb.media.DVBMediaSelectControl.selectServiceMediaComponent(Locator l)`
  can present analog services.

  The interface `org.dvb.media.VideoPresentationControl` and its methods will respond to analog services as well as digital services; the response will be dependent on individual implementations and may vary between platforms.

7.2.1.5.2.1.6 Analog Sources

`javax.tv.media.AWTSizeControl` and its associated methods (e.g., `getSize()`, `setSize()`…) are available to the application developer for all sources: analog and digital.

The method `javax.tv.service.Service.retrieveDetails(SIRequestor requestor)` will return `SIRequestor.notifySuccess` with `DATA_UNAVAILABLE` (if SI was available) for analog services. Result for analog services is `SIRequestor.notifyFailure` with `DATA_UNAVAILABLE` (if SI was not available).

7.2.1.5.2.1.7 Stereoscopic 3D Video APIs (informative)

Packages `org.ocap.media` (Annex S), `org.ocap.service` (Annex P), and `org.ocap.diagnostics` (Annex Z) provide APIs for detection of stereoscopic 3D (S3D) video content and associated presentation issues.

Applications that wish to detect changes to S3D content signaling may subscribe as listeners to `org.ocap.media.VideoFormatControl` and receive notification of the current S3D formatting data via `org.ocap.media.S3DSignalingChangedEvent`.

When presenting S3D content via the Java TV service selection API (see Section 13.3.9), the OCAP implementation indicates detected errors in S3D video presentation via `org.ocap.service.S3DAlternativeContentErrorEvent`. Upon receiving this event, applications may display appropriate warnings and notifications to the user.

**NOTE:** The Host device may be configured to display its own warning messages upon detection of incompatibilities between selected S3D video and the destination display device, according to [HOST] and [MIB-HOST].

In order to determine the 3D content formats supported by the display device, applications may use the `org.ocap.diagnostics.MIBManager` API to query for the `ocStbHostDVHDMISupported3DStructures` MIB object defined in [MIB-HOST].

Applications should not attempt to present S3D video via standalone JMF Players.

7.2.1.5.2.1.8 Interlaced Video Detection

Applications may determine whether input video is interlaced by using the `getScanMode()` method of the `VideoFormatControl` interface in the `org.ocap.media` package (Annex S).
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 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. The priority scheme in Table 10–3 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 10–3). 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 1.0.2] Section: 6 Transport protocols and are extensions of [DVB-MHP 1.0.3] Section: 6 Transport Protocols.

This section covers the transport protocols that OCAP supports.

The OCAP transport protocols enable an OCAP application to communicate with the external world. The OCAP 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 1.0.2] Section. Section 8, Transport Protocols, (this section) of OCAP corresponds to Section 6 of [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2] as follows:

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>MHP Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>8Transport Protocols</td>
<td>6 Transport protocols</td>
<td>Extension</td>
<td>6 Transport Protocols</td>
<td>Extension</td>
</tr>
<tr>
<td>8.1 DVB-GEM and DVB-MHP Specification Correspondence</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>8.2 OCAP Specific Requirements</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>8.2.1 Deviations from DVB-MHP</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>8.2.1.1 Introduction</td>
<td>6.1 Introduction</td>
<td>Extension</td>
<td>6.1 Introduction</td>
<td>Extension</td>
</tr>
<tr>
<td>8.2.1.2 Conditional Access</td>
<td>6.2 Broadcast channel protocols</td>
<td>Extension</td>
<td>6.2 Broadcast Channel Protocols</td>
<td>Extension</td>
</tr>
<tr>
<td>8.2.1.3 MPEG-2 Transport Stream</td>
<td>6.2.1 MPEG-2 transport stream</td>
<td>Compliance</td>
<td>6.2.1 MPEG-2 Transport Stream</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>6.2.2 MPEG-2 sections</td>
<td>Compliance</td>
<td>6.2.2 MPEG-2 Sections</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>6.2.3 DSM-CC private data</td>
<td>Compliance</td>
<td>6.2.3 DSM-CC Private Data</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>6.2.4 DSM-CC data carousel</td>
<td>Compliance</td>
<td>6.2.4 DSM-CC Data Carousel</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>6.2.5 Object carousel</td>
<td>Compliance</td>
<td>6.2.5 DSM-CC User-to-User Object Carousel</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>6.2.6 Protocol for delivery of IP multicast over the broadcast channel</td>
<td>Compliance</td>
<td>6.2.6 DVB Multiprotocol Encapsulation</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>6.2.7 Internet Protocol (IP)</td>
<td>Compliance</td>
<td>6.2.7 Internet Protocol (IP)</td>
<td>Compliance</td>
</tr>
<tr>
<td>OCAP</td>
<td>[DVB-GEM 1.0.2] Section</td>
<td>GEM Compliance</td>
<td>[DVB-MHP 1.0.3] Section</td>
<td>MHP Compliance</td>
</tr>
<tr>
<td>------</td>
<td>------------------------</td>
<td>----------------</td>
<td>--------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>6.2.8 User Datagram Protocol (UDP)</td>
<td>Compliance</td>
<td>6.2.8 User Datagram Protocol (UDP)</td>
<td>Compliance</td>
</tr>
<tr>
<td>8.2.1.4 Service Information</td>
<td>6.2.9 Service information</td>
<td>Extension</td>
<td>6.2.9 DVB Service Information</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>6.2.10 IP signaling</td>
<td>Compliance</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>6.3 Interaction channel protocols</td>
<td>A subsection is extended</td>
<td>6.3 Interaction Channel Protocols</td>
<td>A subsection is extended</td>
</tr>
<tr>
<td>8.2.1.5 Network Dependent Protocols</td>
<td>6.3.1 Network Dependent Protocols</td>
<td>Extension</td>
<td>6.3.1 Network Dependent Protocols</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>6.3.2 Internet Protocol</td>
<td>Compliance</td>
<td>6.3.2 Internet Protocol (IP)</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>6.3.3 Transmission Control Protocol</td>
<td>Compliance</td>
<td>6.3.3 Transmission Control Protocol (TCP)</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>6.3.4 UNO-RPC</td>
<td>Compliance</td>
<td>6.3.4 UNO-RPC</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>6.3.5 UNO-CDR</td>
<td>Compliance</td>
<td>6.3.5 UNO-CDR</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>6.3.6 DSM-CC User to User 17</td>
<td>Compliance</td>
<td>6.3.6 DCM-CC User to User</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>6.3.7 HyperText Transfer Protocol (HTTP)</td>
<td>Compliance</td>
<td>6.3.7 HyperText Transfer Protocol (HTTP)</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>6.3.8 Service Specific</td>
<td>Compliance</td>
<td>6.3.8 Service Specific</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>6.3.9 User Datagram Protocol</td>
<td>Compliance</td>
<td>6.3.9 User Datagram Protocol (UDP)</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>6.3.10 DNS</td>
<td>Compliance</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>8.2.1.6 Multiprotocol Encapsulation</td>
<td>11.5.2 Support for Multicast IP over the Broadcast Channel</td>
<td>Extension</td>
<td>11.5.2 Support for Multicast IP over the Broadcast Channel</td>
<td>Extension</td>
</tr>
<tr>
<td>8.2.1.7 Multicast Support</td>
<td>11.5.3 Support for IP over the return channel</td>
<td>Extension</td>
<td>11.5.3 Support for IP over the return channel</td>
<td>Extension</td>
</tr>
<tr>
<td>8.2.2 Extensions to DVB-GEM (Normative)</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>8.2.2.1 Broadcast Channel Protocols</td>
<td>Section 6.2 Broadcast Channel Protocols</td>
<td>Extension</td>
<td>6.2 Broadcast Channel Protocols</td>
<td>Extension</td>
</tr>
<tr>
<td>8.2.2.2 Interaction Channel Protocols</td>
<td>Section 6.3 Interaction Channel Protocols</td>
<td>Extension</td>
<td>6.3 Interaction Channel Protocols</td>
<td>Extension</td>
</tr>
</tbody>
</table>

Section 6 of the [DVB-GEM 1.0.2] 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 1.0.2] and [CCIF] and extended in this specification.
8.2 OCAP Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

8.2.1 Deviations from DVB-MHP

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

8.2.1.1 Introduction

This subsection contains requirements that do not correspond to any [DVB-GEM 1.0.2] Section: 6.1 Introduction and are extensions of [DVB-MHP 1.0.3] Section: 6.1 Introduction.

The network-dependent protocols for the interaction channels in the OpenCable context are specified in [CCIF] 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 1.0.2] Section: 6.2 Broadcast channel protocols and are extensions of [DVB-MHP 1.0.3] 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, 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 1.0.2] Section: 6.2.1 MPEG-2 transport stream and are extensions of [DVB-MHP 1.0.3] Section: 6.2.1 MPEG-2 Transport Stream.

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

8.2.1.4 Service Information

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 6.2.9 Service information and are extensions of [DVB-MHP 1.0.3] Section: 6.2.9 DVB Service Information.

OCAP 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 platforms SHALL NOT expose information relating to application-type virtual channels through the Service Information interfaces described in Annex T. OCAP 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]:

- stream_identifier_descriptor. This descriptor is used to resolve references to a component, where the reference specifies a component_tag value. For example, component_tag is used in:
• 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-MHP 1.0.3] 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.

OCAP complies with [DVB-MHP 1.0.3] annex B.3.2 "DSM-CC association_tags to DVB component_tags".
• private_data_specifier_descriptor. This descriptor is used in the AIT as defined in [DVB-MHP 1.0.3] Section 10.4.7 "Use of private descriptors in the AIT".
• data_broadcast_id_descriptor. This descriptor syntax is extended in [DVB-MHP 1.0.3] Section 10.7.2.2 "MHP data broadcast id descriptor". Its syntax and usage is defined in that section.

8.2.1.5 Network Dependent Protocols

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 6.3.1 Network Dependent Protocols and are extensions of [DVB-MHP 1.0.3] Section: 6.3.1 Network Dependent Protocols.

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

8.2.1.6 Multiprotocol Encapsulation

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 11.5.2 Support for Multicast IP over the Broadcast Channel and are extensions of [DVB-MHP 1.0.3] 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-MHP 1.0.3] 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 1.0.2] Section: 11.5.3 Support for IP over the Return Channel and are extensions of [DVB-MHP 1.0.3] 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.

NOTE: If IPv4 is supported, then the underlying operating system requires support for IGMPv2 (RFC 2236). Where IPv6 is supported, the equivalent standard is MLD (RFC 2710). At the time, java.net.MulticastSocket does
not support source-specific multicast, so support for the updated RFCs (IGMPv3 and MLDv2) is not required.

8.2.2 Extensions to DVB-GEM (Normative)

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

The following information is normative for OCAP. It is specific to OCAP and is not contained within the [DVB-MHP 1.0.3].

8.2.2.1 Broadcast Channel Protocols

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 6.2 Broadcast Channel Protocols and are extensions of [DVB-MHP 1.0.3] Section: 6.2 Broadcast Channel Protocols.

OCAP SHALL require the following additional broadcast channel protocols.

8.2.2.1.1 VBI

The OCAP implementation SHALL support VBI as specified in [CEA-608-E], 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 1.0.2] Section: 6.3 Interaction Channel Protocols and are extensions of [DVB-MHP 1.0.3] Section: 6.3 Interaction Channel Protocols.

OCAP SHALL require the following interaction channel protocols.

8.2.2.2.1 HDNI

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

8.2.2.2.2 CableCARD

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

8.2.2.2.3 HTTP 1.1

The OCAP implementation SHALL support HTTP 1.1 over the Interaction Channel as defined in Section 7.1 of [ATSC A/96], which extends [DVB-MHP 1.0.3] Section: 15 Detailed Platform Profile Definitions, Table 65 where HTTP 1.1 support is optional.

The OCAP implementation SHALL support HTTPS over the Interaction Channel as defined in Section 9.2 of [ATSC A/96].

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

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] or [DVB-MHP 1.0.3] requirements. OCAP complies with Sections 6.4.1.1, 6.4.1.3, and 6.4.1.4 of [DVB-MHP 1.1.3] for the purpose of defining a file system implemented via the interaction channel as modified below.

Where this definition is applied to the interpretation of a transport protocol with protocol ID 0x0101, [DVB-MHP 1.1.3] Table-1 in Section 6.4.1.1 SHALL be considered to be replaced with the following table:

<table>
<thead>
<tr>
<th>URL byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;<a href="http://www.dvb.org/applications/application1.jar">http://www.dvb.org/applications/application1.jar</a>&quot;</td>
</tr>
<tr>
<td>&quot;<a href="http://www.dvb.org/graphics">http://www.dvb.org/graphics</a>&quot;</td>
</tr>
<tr>
<td>&quot;<a href="http://www.dvb.org/shared/utils.jar">http://www.dvb.org/shared/utils.jar</a>&quot;</td>
</tr>
<tr>
<td>&quot;<a href="http://www.ebu.ch/general/misc.zip">http://www.ebu.ch/general/misc.zip</a>&quot;</td>
</tr>
<tr>
<td>&quot;<a href="http://www.dvb.org/other_stuff/we_dont_use_this_very_often.zip">http://www.dvb.org/other_stuff/we_dont_use_this_very_often.zip</a>&quot;</td>
</tr>
</tbody>
</table>

8.2.2.2.4.1 Authentication of ZIP files

As an alternative to including OCAP security files within the directory structure contained within a ZIP file, the ZIP file MAY be signed and authenticated as a single file. Where a ZIP file is signed according to the OCAP security model, all classes or files to be loaded from that ZIP file SHALL be considered to be signed by the certificates signing the ZIP file under the OCAP security model. That is, when security files for the ZIP file are present and sufficient to authenticate the ZIP file, further authentication of files contained therein is not necessary.

In such a case, the URLs used to locate OCAP security files SHALL be constructed by replacing the final path segment (i.e., the filename) with that of the appropriate OCAP security file name. For example, the hashfile for the directory containing "http://192.168.0.123/app/App.zip" would be "http://192.168.0.123/app/ocap.hashfile".

**NOTE:** Since the security files used to authenticate the ZIP file would be outside the file system composed of the ZIP file, they would not fall under consideration for storage as part of an application. It would be the implementation's responsibility to ensure that the files necessary to authenticate a ZIP when the contained application is stored are maintained in order to satisfy subsequent authentication requirements.
9 CONTENT FORMATS

This section contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 7, Content formats, and are extensions of [DVB-MHP 1.0.3] Section: 7, Content formats.

This section specifies the minimal set of content formats that SHALL be supported by an OCAP-compliant implementation. It also covers the color representation for the image data. Finally, this section identifies compatible MIME types which assist an OCAP 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 1.0.2] Section. Section 9, Content Formats, (this section) of OCAP corresponds to Section 7 of [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2] as follows:

| Table 9–1 - Correlation between OCAP and [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3] |
|-----------------|-----------------|----------------|-----------------|----------------|
| OCAP            | [DVB-GEM 1.0.2]  |                  | [DVB-MHP 1.0.3]  |                  |
|                 | Section         | GEM Compliance  | 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 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 |
| 7.1.1 Bitmap image formats | Compliance | 7.1.1 Bitmap image formats | Compliance |
| 7.1.2 MPEG-2 I-Frames | Compliance | 7.1.2 MPEG-2 I-Frames | Compliance |
| 7.1.3 MPEG-2 Video "drips" | Compliance | 7.1.3 MPEG-2 Video "drips" | Compliance |
| 7.1.4 Monomedia format for audio clips | Compliance | 7.1.4 Monomedia format for audio clips | Extension |
| 7.1.5 Monomedia format for text | Compliance | 7.1.5 Monomedia format for text | Compliance |
| 7.2 Broadcast streaming formats | Compliance | 7.2 Broadcast streaming formats | Compliance |
| 7.2.1 Audio | Compliance | 7.2.1 Audio | No Corresponding Section |
| 7.2.2 Video | Compliance | 7.2.2 Video | No Corresponding Section |
| 7.2.3 Subtitles | Compliance | 7.2.3 Subtitles | Compliance |
9.2 OCAP Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

9.2.1 Deviations from the DVB-MHP Specification

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

9.2.1.1 Audio: Monomedia Format for Audio Clips

This subsection is compliant with [DVB-GEM 1.0.2] Section: 7.1.4 Monomedia format for audio clips and are extensions of [DVB-MHP 1.0.3] Section: 7.1.4 Monomedia format for audio clips.

This section corresponds to Section 7.1.4 of [DVB-GEM 1.0.2] 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/53].

Each file of audio content is a binary data file carrying Audio elementary stream data. Each file delivers an integer number of audio access units that MAY be preceded by arbitrary data of up to 4096 bytes in length and MAY be followed by arbitrary data up to the end of the file. The MPEG Audio data in all other respects SHALL conform to the specifications provided in [ETSI TR 101 154]. The AC-3 Audio data in all other respects SHALL conform to the specifications provided in [ATSC A/53].

The format of the other arbitrary data that may precede or trail the audio access units in a file is unspecified; however, the allowance of such is intended to provide for commonly used tagging mechanisms. An implementation SHALL be able to skip up to 4096 bytes of such data prior to the initial audio access unit, synchronizing on synchronization bytes as described in the relevant audio format specification (see 6.1.2 of [ATSC A/52B] and 2.4.3 of [ISO 11172-3]).

9.2.1.2 Audio: Broadcast Streaming Format

This subsection is compliant with [DVB-GEM 1.0.2] Section: 7.2.1 Audio.

This section corresponds to Section 7.2.1 of [DVB-GEM 1.0.2] as follows:

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

Audio formats SHALL be supported as defined in [OCCEP].
9.2.1.3 Video: Broadcast Streaming Format

This subsection is compliant with [DVB-GEM 1.0.2] Section: 7.2.2 Video.

This section corresponds to Section 7.2.2 of [DVB-GEM 1.0.2] as follows:

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

Video formats SHALL be supported as defined in [OCCEP].

9.2.1.4 MIME Types

This subsection is compliant with [DVB-GEM 1.0.2] Section: 7.6 MIME Types and are extensions of [DVB-MHP 1.0.3] Section: 7.6 MIME Types. This section corresponds to Section 7.6 of [DVB-GEM 1.0.2] as follows:

The table of defined MIME types SHALL be considered extended with following:

<table>
<thead>
<tr>
<th>MIME Type</th>
<th>File Type</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;audio/mpeg&quot;</td>
<td>&quot;.mp3&quot;</td>
<td>As defined in 9.2.1.1</td>
</tr>
<tr>
<td>&quot;audio/ac3&quot;</td>
<td>&quot;.ac3&quot;</td>
<td>As defined in 9.2.1.1</td>
</tr>
</tbody>
</table>

9.2.2 Extensions to DVB-GEM

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

The following information extends OCAP. It attempts to clarify issues in the [DVB-GEM 1.0.2] as well as provide additional information to the normative requirements.

9.2.2.1 Closed-Captioning

OCAP SHALL support closed-captioning as specified in Section 8.2.4 of [HOST].

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

9.2.3 OOB/DSG Object Carousel

OCAP 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 should not expect that all networks and CableCARDs will implement support for the OOB object carousel.

9.2.4 OpenType Font Support

OCAP SHALL support the OpenType downloadable font as defined in section 7.4.2 of [DVB-MHP 1.1.3].
10 APPLICATION MODEL

This section contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 9 Application model and are extensions of [DVB-MHP 1.0.3] Section: 9 Application model.

The application model defines the control of the lifecycle of OCAP applications. OCAP 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 extends the DVB-MHP application model via the introduction of unbound applications. Unbound applications and Monitor Applications are broadcast service independent. OCAP 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 1.0.2] Section. Section 10 (this section) of OCAP corresponds to Section 9 of [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2] as follows:

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>MHP Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 Application Model</td>
<td>9 Application model</td>
<td>Extension</td>
<td>9 Application model</td>
<td>Extension</td>
</tr>
<tr>
<td>10.1 DVB-GEM and DVB-MHP Specification Correspondence</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>10.2 OCAP Specific Requirements</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>10.2.1 Deviations from the DVB-MHP Specification</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>10.2.1.1 General</td>
<td>9 Application model</td>
<td>Extension</td>
<td>9 Application model</td>
<td>Extension</td>
</tr>
<tr>
<td>10.2.1.2 Application Model of Bound Application</td>
<td>9.1 Broadcast GEM application</td>
<td>Extension</td>
<td>9.1 Broadcast MHP applications</td>
<td>Extension</td>
</tr>
<tr>
<td>10.2.1.3 OCAP-J Model</td>
<td>9.2 DVB-J model</td>
<td>Extension</td>
<td>9.2 DVB-J model</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>9.3 DVB-HTML model</td>
<td>Compliance</td>
<td>9.3 DVB-HTML model</td>
<td>Compliance</td>
</tr>
<tr>
<td>10.2.1.4 Inter-application Resource Management</td>
<td>9.4 Inter-application resource management</td>
<td>Extension</td>
<td>9.4 Inter-application resource management</td>
<td>Extension</td>
</tr>
<tr>
<td>10.2.2 Extensions to DVB-GEM (Normative)</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
</tbody>
</table>
10.2 OCAP Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

10.2.1 Deviations from the DVB-MHP Specification

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

10.2.1.1 General

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 9 Application model and are extensions of [DVB-MHP 1.0.3] Section: 9 Application model.

All instances of the term "MHP" in the Section 9 Application model and its sub-sections of [DVB-GEM 1.0.2] 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 1.0.2] 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 1.0.2] SHALL be replaced by "bound application".

The term "navigator" in the Section 9 Application model and its sub sections of [DVB-GEM 1.0.2] SHALL be replaced by "unbound application".

OCAP-J application types MAY make use of any OCAP API as permissions allow. All applications in OCAP 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 1.0.2] Section: 9.1 Broadcast GEM application and are extensions of [DVB-MHP 1.0.3] Section: 9.1 Broadcast MHP applications.

The bound application of OCAP corresponds to the broadcast MHP application. OCAP extends the application model of the bound application as defined in [DVB-GEM 1.0.2] 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 1.0.2] is replaced by "bound application". OCAP 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-MHP 1.0.3], Section 9.1.1 Basic lifecycle.

10.2.1.3 OCAP-J Model

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 9.2 DVB-J model and are extensions of [DVB-MHP 1.0.3] Section: 9.2 DVB-J Model.

10.2.1.4 Inter-application Resource Management

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 9.4 Inter-application resource management and are extensions of [DVB-MHP 1.0.3] 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 1.0.2] Section.

The following information is normative for OCAP. It is specific to OCAP and is not contained within the [DVB-MHP 1.0.3].

10.2.2.1 OCAP Applications

In [DVB-MHP 1.0.3] all applications are related to a service and their lifecycle is linked to the currently selected broadcast service. OCAP introduces applications that are not linked to the broadcast service and whose lifecycle is independent of the AIT signaling. OCAP 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 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 application modes. These modes are defined in Section 10.2.2.7.

This section describes the extensions to the [DVB-GEM 1.0.2] broadcast service and application lifecycle model that are available in OCAP to manage these types of applications.

10.2.2.2 OCAP Service Model

10.2.2.2.1 Service Context Creation and Destruction

OCAP 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.
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 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 (a) and (b) 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 (c) is immutable between boot processes; thus applications signaled in the XAIT or defined in an 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 an 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 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 an 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

<table>
<thead>
<tr>
<th>Application_information_section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table_id = 0x74</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>common_descriptors_length = ***</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>abstract_service_descriptor {</td>
</tr>
<tr>
<td>descriptor_tag = 0x66</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>service_id = 0x020001</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>service_name_byte = service1</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>abstract_service_descriptor_ {</td>
</tr>
<tr>
<td>descriptor_tag = 0x66</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>service_id = 0x020002</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>service_name_byte = service2</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Reserved_future_use</td>
</tr>
<tr>
<td>application_loop_length = ***</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>application_identifier {</td>
</tr>
<tr>
<td>organization_id = 1</td>
</tr>
<tr>
<td>application_id = 1</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>application_descriptor_loop_length = ***</td>
</tr>
<tr>
<td>unbound_application_descriptor {</td>
</tr>
<tr>
<td>descriptor_tag = 0x67</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>service_id = 0x020001</td>
</tr>
<tr>
<td>version_number = 0</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>application_identifier {</td>
</tr>
<tr>
<td>organization_id = 1</td>
</tr>
<tr>
<td>application_id = 2</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>application_descriptor_loop_length = ***</td>
</tr>
<tr>
<td>unbound_application_descriptor {</td>
</tr>
<tr>
<td>descriptor_tag = 0x67</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>service_id = 0x020002</td>
</tr>
<tr>
<td>version_number = 0</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Services</th>
<th>Associated Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>service1 (service_id = 0x020001)</td>
<td>{application_id = 1, organization_id = 1}</td>
</tr>
<tr>
<td>service2 (service_id = 0x020002)</td>
<td>{application_id = 2, organization_id = 1}</td>
</tr>
</tbody>
</table>

Step 3. 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.

<table>
<thead>
<tr>
<th>Services</th>
<th>Associated Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>service2 (service_id = 0x020002)</td>
<td>{application_id = 3, organization_id = 1}</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Services</th>
<th>Associated Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>service1 (service_id = 0x020001)</td>
<td>{application_id = 1, organization_id = 1}</td>
</tr>
<tr>
<td>service2 (service_id = 0x020002)</td>
<td>{application_id = 2, organization_id = 1}</td>
</tr>
</tbody>
</table>

Step 3. 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.

<table>
<thead>
<tr>
<th>Services</th>
<th>Associated Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>service2 (service_id = 0x020002)</td>
<td>{application_id = 3, organization_id = 1}</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Services</th>
<th>Associated Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>service1 (service_id = 0x020001)</td>
<td>{application_id = 1, organization_id = 1}</td>
</tr>
<tr>
<td>service2 (service_id = 0x020002)</td>
<td>{application_id = 2, organization_id = 1}</td>
</tr>
</tbody>
</table>

Step 3. 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.

<table>
<thead>
<tr>
<th>Services</th>
<th>Associated Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>service1 (service_id = 0x020001)</td>
<td>{application_id = 1, organization_id = 1}</td>
</tr>
<tr>
<td>service2 (service_id = 0x020002)</td>
<td>{application_id = 2, organization_id = 1}</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Services</th>
<th>Associated Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>service1 (service_id = 0x020001)</td>
<td>{application_id = 1, organization_id = 1}</td>
</tr>
<tr>
<td>service2 (service_id = 0x020002)</td>
<td>{application_id = 2, organization_id = 1}</td>
</tr>
<tr>
<td></td>
<td>{application_id = 3, organization_id = 1}</td>
</tr>
</tbody>
</table>

10.2.2.2.2 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 a javax.tv.service.selection.SelectionFailedEvent to be generated.

The OCAP implementation SHALL allow applications within an abstract service to be executed and presented concurrently as indicated by the application signaling. Additionally, the OCAP 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-MHP 1.0.3] 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(javax.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-MHP 1.0.3], 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.
10.2.2.3.3 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 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("*", "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 10.2.2.3.1 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
application with MonitorAppPermission("servicemanager") also has AppsControlPermission as defined
in [DVB-MHP 1.0.3] to control the life cycles of applications in its own service context.

10.2.2.4 Broadcast Service Selection

The [DVB-MHP 1.0.3] 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 is fully compliant with [DVB-MHP 1.0.3] (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 10.2.2.3.3, 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-MHP 1.0.3] and this is the subject of an unresolved
issue against that specification.

10.2.2.5 Switched Service Management

The [DVB-MHP 1.1.3] specification provides a service provider interface (SPI) that allows a privileged application
to control a set of services based on a scheme, e.g., "ocap://". OCAP, this specification, complies with the SPI in
[DVB-MHP 1.1.3] Annex AN. This section extends the SPI for handling of switched digital services.

Application usage of the SPI is defined in the org.dvb.spi package. Using this definition, an application can modify
service tuning information before the service is selected. This allows a performance optimization where the provider
does not need to be called by the implementation during a service selection. The implementation is still required to
call the org.dvb.spi.selection.SelectionProvider.newSession method for services the
SelectionProvider registered interest in. When the implementation does not solicit the SelectionProvider for tuning
information, it MAY call the newSession method after service selection has completed for those services.

When the implementation solicits the SelectionProvider for tuning information via the
SelectionSession.select method, the lifetime of that tuning information SHALL be limited to the lifetime of
that SelectionSession. That is, the tuning information returned from SelectionSession.select SHALL
NOT continue to be used after the SelectionSession.destroy method is invoked on that SelectionSession
unless the same tuning information has otherwise been provided by another active SelectionSession or specified
via a provider-created KnownServiceReference.
A SelectionProvider application MAY include in the set of services it returns from the `getServiceList` or `serviceListChanged` methods any service created using the `org.javax.tv.service.SIManager.getService` or `filterService` methods. Section 16.2.1.5.3 describes how an application MAY create a service using the `org.javax.tv.service.SIManager.getService` or `filterService` methods.

When `MonitorAppPermission("servicemanager")` is granted to a privileged application, it SHALL imply `org.dvb.spi.ProviderPermission("*", "system")` is also granted; see the `java.security.BasicPermission.implies` method. This provides a privileged application with the ability to register as a selection provider with the `org.dvb.spi.ProviderRegistry`.

When a privileged application updates tuning information for a switched service (via calls to the `org.dvb.spi.selection.SelectionProviderContext.updateService` method), any current activity involving the service SHALL be updated with the new information, and any activity involving the service for which tuning was implied SHALL effect an implicit re-tune using the updated information. Such implicit re-tunes SHALL NOT be visible via the `org.davic.net.tuning` API. Note that support for switched services is not limited to service selection via `ServiceContext`, but applies to any valid use or reference to a switched digital service. This includes, for example, tuning via `NetworkInterfaceController`.

It is an implementation option whether to maintain a single `SelectionSession` or multiple `SelectionSession`s for multiple concurrent activities involving a switched service. Provider applications SHOULD be prepared for either case.

Upon un-registration of a `SelectionProvider`, all objects and information previously supplied by that `SelectionProvider` SHALL be forgotten by the implementation. This includes updates to pre-existing services signaled in the VCT, as well as services defined entirely by the provider. Destruction of the provider application SHALL imply un-registration.

Section 16.2.1.6 definition of transport independent and dependent locators necessitates clarification of method implementation in the provider SPI complied with in [DVB-MHP 1.1.3] Annex AN. The following rules comprise the clarification:

- `org.dvb.spi.selection.SelectionProvider.getServiceList` method: The implementation SHALL ignore any `ServiceReference` objects returned by the method that do not contain correctly formatted locators as defined in Section 16.2.1.6. The sentence in the method javadoc, "Where the transport independent identification of a service is equal to one already in the service list then that transport independent service shall acquire an additional transport dependent service," SHALL be read as "Where the transport dependent identification of a service is equal to one already in a service list returned in a previous call to this method or in the implementation SI database, then that transport dependent service SHALL be updated with the actual locator if one is provided in the `ServiceReference` returned by this method."

- `org.dvb.spi.selection.SelectionProviderContext.serviceDescriptionAvailable`: The implementation SHALL ignore any `ServiceReference` objects returned by the method that do not contain correctly formatted locators as defined in Section 16.2.1.6.

- `org.dvb.spi.selection.SelectionProviderContext.serviceListChanged`: The implementation SHALL ignore any `ServiceReference` objects returned by the method that do not contain correctly formatted locators as defined in Section 16.2.1.6.

**10.2.2.3 Application Signaling and Lifecycle**

The OCAP-J application lifecycle differs from that described in [DVB-MHP 1.0.3] in the following way:

The term "application" in Section 9.2.3 of [DVB-MHP 1.0.3] 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-MHP 1.0.3]). In
the case that the OCAP-J application is signaled as AUTO_START, the rules described in Section 9.1.6 of [DVB-MHP 1.0.3] apply and a new OCAP-J application instance is not auto-started when the application returns to the unloaded state. When an application signaled as AUTO_START and protocol_id 0x101, is loading, and any file download failure prevents it from transitioning to the Active state, then upon subsequent receipt of the same table in which the application was signaled, i.e., AIT or XAIT with matching version number, the implementation SHALL return the application to new auto-start status as defined by Section 9.1.6 of [DVB-MHP 1.0.3]. This will cause the application to be re-launched at that time.

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-MHP 1.0.3] 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-MHP 1.0.3], `org.dvb.application.CurrentServiceFilter` and `org.dvb.application.RunningApplicationsFilter` class descriptions SHALL be read as "any current service".

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:

1. 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.

2. 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 and stored (see OCAPAppAttributes.isNewVersionSignaled() and OCAPAppAttributes.hasNewVersion()); 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 (both signaled and/or stored).

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

Each environment is always in one of four states:

- Selected - See Selected Environment in Section 3.2.1.
- Background - Environment is not selected.
- Presenting - In some situations, applications might be presenting a User Interface but its home environment is not selected and is not setting the rules and policies of the device. For instance, for devices that support a Picture-in-Picture (PiP) feature, a viewer might choose to watch a DVD on the main screen while watching cable TV in a PiP window. In this scenario, the DVD might be the selected environment, and cable might be in the presenting state.
- Inactive - An environment might be a capability of a device but is currently not supporting any executing applications. For instance, the DVD environment might not be active unless a DVD disc is inserted.

OCAP host devices SHALL support a cable environment as described in the specification for the class org.ocap.environment.Environment (Annex Y). OCAP host devices MAY also support one or more non-cable environments as described in Annex A. OCAP host devices that 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 for the purpose of changing the selected state. However, the application_mode_descriptor SHALL be retained so that org.ocap.application Interface OcapAppAttributes.getApplicationMode() can subsequently return the applicable associated integer value.

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

Table 10–2 summarizes the combinations of environments and environment states suitable for common conditions found in an OCAP host device that supports a cable environment and a manufacturer environment.
### Table 10–2 - Combinations of environments and states

<table>
<thead>
<tr>
<th>Condition</th>
<th>Manufacturer environment</th>
<th>Cable environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inactive manufacturer environment, e.g., when no media is inserted</td>
<td>inactive</td>
<td>active</td>
</tr>
<tr>
<td>No CableCARD inserted</td>
<td>selected</td>
<td>inactive</td>
</tr>
<tr>
<td>End-user is interacting with OCAP applications</td>
<td>background</td>
<td>selected</td>
</tr>
<tr>
<td>End-user is interacting with non-OCAP applications</td>
<td>selected</td>
<td>background</td>
</tr>
<tr>
<td>End-user is interacting with OCAP applications; non-OCAP applications remain visible in separate &quot;window&quot;</td>
<td>presenting</td>
<td>selected</td>
</tr>
<tr>
<td>End-user is interacting with non-OCAP applications; OCAP applications remain visible in separate &quot;window&quot;</td>
<td>selected</td>
<td>presenting</td>
</tr>
</tbody>
</table>

OCAP host devices that support multiple environments SHALL support at a minimum the combinations of environments in the selected and background states as well as for the "no CableCARD inserted" condition outlined in Table 10–2. Such devices MAY maintain environment visibility (perhaps in scaled form) even when the user is not interacting with a given environment. For example, if the environment supports PiP, then the environment might be in the secondary smaller window. When an environment and its component applications are visible but is not the primary focus of user interaction, then that environment SHALL be in the presenting state.

When a host boots, the first environment selected is addressed in Section 20.2.1.

#### 10.2.2.4.1 Initiation of state transitions

Environments become selected as a consequence of choices made by the end user. The following scenarios are possible:

- 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 is 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 OCAP or host device manufacturer applications using some kind of application manager UI. Normally, choosing an OCAP application results in the cable environment becoming selected; selecting a non-OCAP application results in a non-cable environment. The exception to this is choosing an application that 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 require 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 a host should automatically start an application to handle that content. Such applications could be OCAP applications (e.g., a photo viewer) or non-OCAP 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-user's chosen application.

Those hosts that support simultaneous display of two or more environments (e.g., in a Picture-in-Picture configuration) SHALL ensure that the selected environment reflects the primary focus of user interaction. Only one environment is to be the primary focus of user interaction at a time, just as only one environment may be selected at a time.

10.2.2.4.2 Introduction (informative)

Host device manufacturer applications are OCAP applications deployed to the host by means that are outside the scope of OCAP. A non-OCAP application can never access the embedded cable return channel (i.e., SocketPermissions can never be granted) and is limited to functioning as a cross-environment or background application when the cable environment is selected. Implementers of host device manufacturer 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.

Host device manufacturer applications may be written in any language, including (but not limited to) C, C++ or Java, as long as all requirements of the OCAP specification are complied with.

Host device manufacturer applications 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 host device manufacturer 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, other 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 environments associated with non-OCAP applications and environments, e.g., a Blu-ray environment that runs BD-J applications. A method in both the Blu-ray and OCAP specifications may have a single implementation that detects the type of the calling application and complies with the appropriate specification for that application.

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.

- Host device manufacturer 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 host device manufacturer 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 determine which resources may be accessed by the application.

10.2.2.4.3 Host Device Manufacturer Applications and the Cable Environment (Applications that access Cable resources)

Host device manufacturer applications SHALL be OCAP applications or execute with an OCAP application front-end if any 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.

Host device manufacturer applications for which none of the above apply MAY be OCAP 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 application front-end. 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 that 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 application front-end 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 a 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 AUTOSTART status SHALL be started when the corresponding abstract service is selected.

f. Application Lifecycle - Host device manufacturer applications SHALL honor any lifecycle commands that are expressed using the AppProxy object that represents the OCAP application front-end.
10.2.2.4.4 Graphics Interactions Between OCAP and non-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 repaint 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.

d. The OCAP implementation SHALL manage the z-ordering of the top level UI containers of both native and OCAP applications such that applications are correctly obscured by the top level UI containers of those other applications in front of them in the z-order.

10.2.2.5 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 applications are described in Table 10–3.

<table>
<thead>
<tr>
<th>Priority Range</th>
<th>Application Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>Initial Monitor Application</td>
</tr>
<tr>
<td>100 to 254</td>
<td>Unbound Application</td>
</tr>
<tr>
<td>1 to 200</td>
<td>Service Bound Application</td>
</tr>
</tbody>
</table>

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.

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.

10.2.2.6 Non-OCAP API Access

OCAP applications MAY access functionality outside the OCAP API, via non-OCAP APIs. Non-OCAP APIs MAY be part of an environment that is proprietary to the Host device. OCAP-J applications that access non-OCAP APIs are not considered interoperable applications, as they depend upon implementation-specific features.

Where OCAP applications use non-OCAP APIs that interact with OCAP-exclusive and shared resources, access to such resources 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 10.2.2.7 describes requirements for host device manufacturer applications that apply equally to native APIs.
10.2.2.7 Application Modes

Applications may exist in one of four modes: normal, cross-environment, background and paused. For applications that 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 that follows the Xlet state machine can only be in the paused state.

The application mode that will be applied to an application when its home environment leaves the selected or presenting states SHALL be determined based upon signaling at the time the application is launched. Changes in application mode signaling SHALL NOT affect running applications or their representation in the applications database.

10.2.2.7.1 Normal Mode

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

10.2.2.7.2 Cross-environment Mode

An application runs in cross-environment mode when its home environment is not selected and it is cross-environment-capable. Xlets in cross-environment mode may be in any state - loaded, paused, active, or destroyed. 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 cable is the selected environment include the following:

- Utility applications such as volume control, settings, input selection
- Reminders set via the manufacturer guid

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 that can run as cross-environment must declare themselves as such via a mechanism described in Section 11.2.2.3.18. If the user selects an application that can run as cross-environment and that application's environment is not the selected one, the application will start 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 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 that 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 that are visible when the selected environment changes and which are then put into 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.

10.2.2.7.3 Background Mode

An application runs in background mode when its home environment is not selected and it is background-capable. Background mode limits the application's ability to interact with the user while limiting the impact on the application itself. Xlets in background mode may be in any state - loaded, paused, active, or destroyed.

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 that 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 unbound applications can be placed into background mode.

10.2.2.7.4 Paused Mode

An application runs in paused mode when its home environment is not selected and it is pauseable. 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 unbound applications can be paused.

Paused mode builds upon the attributes of background mode with the following new application attributes:

- They SHOULD NOT be 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 pauseable 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 that is not empty and an implementation of the startXlet method that 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 (see Section 10.2.2.4), the application provider is responsible for ensuring that these applications will run well when the cable environment is in the background state.
10.2.2.7.5  Legacy Mode

A OCAP application MAY run in legacy mode when the cable environment is not selected and it is not environment aware. Legacy mode not only limits the application's ability to interact, but also limits the effect such environment state transitions have on the application. Legacy applications are effectively isolated from most of the side-effects of environment state transitions that might adversely affect the execution of the application. Legacy mode is considered the default mode for OCAP applications, with the intent of preserving backward-compatibility for applications targeted at previous versions of OCAP.

In general, side-effects of the transition of the home environment to background state will not be visible to legacy applications. While access to a resource may be lost, this is not reported to the application; instead, the connection with the resource is simply broken - temporarily, until the home environment is re-selected.

The following constraints apply to applications running in legacy mode:

• All graphics operations to the screen SHALL fail silently. The state of the application's graphical component hierarchy (rooted with the HScene) SHALL NOT be affected in a way noticeable to the application.

• User input SHALL NOT be delivered to the application.
  • Focus SHALL be revoked and all focus requests ignored.
  • UserEvents, even where the application maintains reservations, SHALL NOT be delivered to the application.

• Access to DAVIC resources MAY be lost; however, the application SHALL NOT be notified of such when the loss is due to the environment state transition. Access to the resource matching the assumed reservation SHALL be restored when the home environment is re-selected. While in legacy mode, the general rule is that methods using such lost resources fail silently. For specific resources where access is removed per selected environment policy, the following legacy mode semantics SHALL apply:
  • HScreenDevice: operations succeed normally, but are not applied until the environment is restored.
  • NetworkInterfaceController (and NetworkInterface): operations appear to succeed; however, subsequent operations that access the transport stream (e.g., section filtering or service presentation) encounter errors as if there is no content.
  • SectionFilterGroup and VBIFilterGroup: filters may be set, but resources are not allocated, and content is not received until access is restored.
  • UserEventRepository: Keys are not delivered.
• The application MAY be starved of processor cycles.
11 APPLICATION SIGNALING

This section contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 10 Application signaling and are extensions of [DVB-MHP 1.0.3] Section: 10 Application Signaling.

This section covers how the OCAP terminal identifies OCAP 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 1.0.2] Section. Section 11 (this section) of OCAP corresponds to Section 10 of [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2] as detailed in Table 11–1.

When reading [DVB-MHP 1.0.3] or [DVB-GEM 1.0.2] with regard to OCAP correspondence, the term "DVB" or "GEM" SHALL be read as "OCAP", and the term "DVB-J" SHALL be read as "OCAP-J".

Table 11–1 - Correlation between OCAP and [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3]

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>MHP Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 Application Signaling</td>
<td>10 Application signaling</td>
<td>Extension</td>
<td>10 Application signaling</td>
<td>Extension</td>
</tr>
<tr>
<td>11.1 DVB-GEM and DVB-MHP</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td>Extension</td>
</tr>
<tr>
<td>Specification Correspondence</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.2 OCAP Specific Requirements</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td>Extension</td>
</tr>
<tr>
<td>11.2.1 Deviations from the</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td>Extension</td>
</tr>
<tr>
<td>DVB-MHP Specification</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.2.1.1 Introduction</td>
<td>10.1 Introduction</td>
<td>Extension</td>
<td>10.1 Introduction</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10.2 Program specific</td>
<td>Compliance</td>
<td>10.2 Program specific</td>
<td>A subsection is extended</td>
</tr>
<tr>
<td>information</td>
<td>information</td>
<td></td>
<td>information</td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10.2 Program specific</td>
<td>Compliance</td>
<td>10.2.1 Application</td>
<td>Compliance</td>
</tr>
<tr>
<td>information</td>
<td>information</td>
<td></td>
<td>signaling stream</td>
<td></td>
</tr>
<tr>
<td>11.2.1.2 Data broadcast streams</td>
<td>10.2 Program specific</td>
<td>Compliance</td>
<td>10.2.2 Data broadcast</td>
<td>Extension</td>
</tr>
<tr>
<td>information</td>
<td>information</td>
<td></td>
<td>streams</td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10.3 Locators within an</td>
<td>Compliance</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>application description</td>
<td>application description</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10.4 Application</td>
<td>Compliance</td>
<td>10.3 Notation</td>
<td>Compliance</td>
</tr>
<tr>
<td>description</td>
<td>description</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10.4 Application</td>
<td>A subsection is extended</td>
<td>10.4 Application</td>
<td>A subsection is extended</td>
</tr>
<tr>
<td>description</td>
<td>Information</td>
<td></td>
<td>Information</td>
<td></td>
</tr>
<tr>
<td>Table</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10.4 Application</td>
<td>Compliance</td>
<td>10.4.1 Data Errors</td>
<td>Compliance</td>
</tr>
<tr>
<td>description</td>
<td>description</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10.4.1 Application</td>
<td>Compliance</td>
<td>10.4.2 AIT</td>
<td>Compliance</td>
</tr>
<tr>
<td>Description Transmission</td>
<td>Description</td>
<td></td>
<td>transmission and</td>
<td></td>
</tr>
<tr>
<td>and Monitoring</td>
<td>Table</td>
<td></td>
<td>monitoring</td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10.4 Application</td>
<td>Compliance</td>
<td>10.4.3 Optimized AIT</td>
<td>Compliance</td>
</tr>
<tr>
<td>Description</td>
<td>description</td>
<td></td>
<td>signaling</td>
<td></td>
</tr>
<tr>
<td>OCAP</td>
<td>[DVB-GEM 1.0.2] Section</td>
<td>GEM Compliance</td>
<td>[DVB-MHP 1.0.3] Section</td>
<td>MHP Compliance</td>
</tr>
<tr>
<td>------</td>
<td>------------------------</td>
<td>----------------</td>
<td>------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10.4.2 Visibility of application description</td>
<td>Compliance</td>
<td>10.4.4 Visibility of AIT</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10.4.3 Content of the application description</td>
<td>Compliance</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10.4 Application description</td>
<td>Compliance</td>
<td>10.4.5 Definition of sub-table for the AIT</td>
<td>Compliance</td>
</tr>
<tr>
<td>11.2.1.3 Syntax of the AIT</td>
<td>10.4 Application description</td>
<td>Compliance</td>
<td>10.4.6 Syntax of the AIT</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10.4 Application description</td>
<td>Compliance</td>
<td>10.4.7 Use of private descriptors in the AIT</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10.4 Application description</td>
<td>Compliance</td>
<td>10.4.8 Text encoding in AIT</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10. Application Signaling</td>
<td>Compliance</td>
<td>10.5 Application identification</td>
<td>A subsection is extended</td>
</tr>
<tr>
<td>11.2.1.10 Application identification</td>
<td>10. Application Signaling</td>
<td>Compliance</td>
<td>10.5.1 Encoding</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10. Application Signaling</td>
<td>Compliance</td>
<td>10.5.3 Authentication of application identification</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10. Application Signaling</td>
<td>Compliance</td>
<td>10.6 Control of application life cycle</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10. Application Signaling</td>
<td>Compliance</td>
<td>10.7 Generic descriptors</td>
<td>Compliance</td>
</tr>
<tr>
<td>No corresponding Section</td>
<td>10. Application Signaling</td>
<td>Compliance</td>
<td>10.8 Transport protocol descriptors</td>
<td>Compliance</td>
</tr>
<tr>
<td>11.2.1.4 Transport Protocol Descriptor</td>
<td>10. Application Signaling</td>
<td>Compliance</td>
<td>10.8.1 Transport protocol descriptor</td>
<td>Extension</td>
</tr>
<tr>
<td>11.2.1.5 Transport via OC</td>
<td>10. Application Signaling</td>
<td>Compliance</td>
<td>10.8.1.1 Transport via OC</td>
<td>Extension</td>
</tr>
<tr>
<td>11.2.1.6 Transport via IP</td>
<td>10. Application Signaling</td>
<td>Compliance</td>
<td>10.8.1.2 Transport via IP</td>
<td>Extension</td>
</tr>
<tr>
<td>11.2.1.7 Transport via Interaction Channel</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10. Application Signaling</td>
<td>Compliance</td>
<td>10.8.3 Pre-fetch signaling</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10.5 DVB-J specific application description</td>
<td>Compliance</td>
<td>10.9 DVB-J specific descriptors</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>10. Application Signaling</td>
<td>Compliance</td>
<td>10.10 DVB-HTML Specific descriptors</td>
<td>Compliance</td>
</tr>
<tr>
<td>11.2.1.8 Constant values</td>
<td>10. Application Signaling</td>
<td>Compliance</td>
<td>10.11 Constant values</td>
<td>Extension</td>
</tr>
<tr>
<td>11.2.1.9 Service Information</td>
<td>10. Application Signaling</td>
<td>Compliance</td>
<td>10.12 Service Information</td>
<td>Extension</td>
</tr>
<tr>
<td>11.2.2 Extensions to DVB-GEM (Normative)</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
</tbody>
</table>

### 11.2 OCAP Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

#### 11.2.1 Deviations from the DVB-MHP Specification

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.
11.2.1.1 Introduction

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 10.1 Introduction and are extensions of [DVB-MHP 1.0.3] Section: 10.1 Introduction.

Signaling is used in OCAP, 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-MHP 1.0.3] and [DVB-GEM 1.0.2] to cover signaling of unbound applications.

OCAP 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; one for service bound applications, the other two 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], Section 9.14.
- OCAP 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-MHP 1.0.3]. Signaling for OCAP unbound applications adheres to OCAP 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 1.0.2] Section: 10.2 Program specific information and are extensions of [DVB-MHP 1.0.3] 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 1.0.2] Section: 10.4 Application description and are extensions of [DVB-MHP 1.0.3] Section: 10.4.6 Syntax of the AIT.

OCAP complies with Section 10.4.6 of the [DVB-MHP 1.0.3] except where stated below.
Application types SHALL be modified so that application_type 0x0001 is described as an OCAP-J application. The application_type 0x0002 DVB-HTML is out of scope of OCAP, therefore, application_type 0x0002...0xFFFF are subject to registration with CableLabs.

11.2.1.4 Transport Protocol Descriptor

This subsection is compliant with [DVB-GEM 1.0.2] Section: 10. Application Signaling and are extensions of [DVB-MHP 1.0.3] Section: 10.8.1 Transport protocol descriptor.

OCAP differs from Section 10.8.1 of the [DVB-MHP 1.0.3] in the values defined for the protocol_id.

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

Protocol_id of [DVB-MHP 1.0.3] is replaced with the following Table to modify reference sections:

<table>
<thead>
<tr>
<th>protocol_id</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>reserved_future_use</td>
</tr>
<tr>
<td>0x0001</td>
<td>OCAP Object Carousel as defined in Section 22</td>
</tr>
<tr>
<td>0x0002</td>
<td>IP via DVB multi protocol encapsulation. Not supported by OCAP.</td>
</tr>
<tr>
<td>0x0003</td>
<td>Reserved for MHP1.1</td>
</tr>
<tr>
<td>0x0004...0x00FF</td>
<td>Reserved_future_use</td>
</tr>
<tr>
<td>0x0100</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0101</td>
<td>See Section 11.2.1.7 for an AIT case, and Section 11.2.2.3.9 for an XAIT case.</td>
</tr>
<tr>
<td>0x0102...0xFFFF</td>
<td>Subject to registration in TR 101 162</td>
</tr>
</tbody>
</table>

Semantic of selector bytes of [DVB-MHP 1.0.3] is replaced with the following Table to modify reference sections:

<table>
<thead>
<tr>
<th>protocol_id</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>reserved_future_use</td>
</tr>
<tr>
<td>0x0001</td>
<td>See Section 11.2.1.5 for an AIT case and Section 11.2.2.3.7 for an XAIT case.</td>
</tr>
<tr>
<td>0x0002</td>
<td>See Section 11.2.2.3.8 for an XAIT case. Not supported.</td>
</tr>
<tr>
<td>0x0003...0x0100</td>
<td>Not defined in this version of the specification.</td>
</tr>
<tr>
<td>0x0101</td>
<td>See Section 11.2.1.7 for an AIT case, and Section 11.2.2.3.9 for an XAIT case.</td>
</tr>
<tr>
<td>0x0102...0xFFFF</td>
<td>Not defined in this version of the specification.</td>
</tr>
</tbody>
</table>

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 1.0.2] Section: 10. Application Signaling and are extensions of [DVB-MHP 1.0.3] Section: 10.8.1.1 Transport via OC.
OCAP differs from Section 10.8.1.1 of [DVB-MHP 1.0.3] 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 with regard to OCAP-J application transmission.

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.

### 11.2.1.6 Transport via IP

This subsection is compliant with [DVB-GEM 1.0.2] Section: 10. Application Signaling and are extensions of [DVB-MHP 1.0.3] Section: 10.8.1.2 Transport via IP.

This section corresponds to Section 10.8.1.2 of [DVB-MHP 1.0.3], OCAP 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 1.0.2] section.

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

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Bits</th>
<th>Mnemonic</th>
</tr>
</thead>
<tbody>
<tr>
<td>for( i=0; i&lt; URL_length; i++) {</td>
<td>8</td>
<td>uimsbf</td>
</tr>
<tr>
<td>URL_byte</td>
<td>8</td>
<td>uimsbf</td>
</tr>
</tbody>
</table>

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

**URL_byte**: These bytes form an HTTP URL to a remote resource 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. The URL SHALL identify either a ZIP file, identified by a ".zip" or ".jar" extension, or a base URL. See Section 8.2.2.2.4 for a description of this file system.

Files to be transferred SHALL be transferred individually by HTTP protocol (i.e., each class file is fetched by a separate HTTP "GET" request).

Files contained within the mounted file system SHALL be accessible via the usual APIs. For example, when an application calls `new java.io.File(".")`, the resultant object SHALL refer to the base directory as provided by the file system.

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 a file system SHALL be listed in the applications database and be launched in the same fashion as any other application so listed.

Multiple transport protocol descriptors with the protocol ID value 0x0101 and the same transport protocol label MAY be provided to define a larger set of URLs to describe the file system. The order in which such transport protocols are entered in the AIT or XAIT SHALL determine the order in which the URLs are searched when locating a file as described in Section 8.2.2.2.4.

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 1.0.2] Section: 10. Application Signaling and are extensions of [DVB-MHP 1.0.3] Section: 10.11 Constant values.

OCAP differs from Section 10.11 of [DVB-MHP 1.0.3] as follows:

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

11.2.1.9 Service Information

This subsection is compliant with [DVB-GEM 1.0.2] Section: 10. Application Signaling and are extensions of [DVB-MHP 1.0.3] Section: 10.12 Service Information.

OCAP differs from Section 10.12 of [DVB-MHP 1.0.3] in how the service identifier descriptor is delivered. In OCAP, 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-MHP 1.0.3] 14.9.1, "Syntax of the textual service identifier".

11.2.1.10 Application identification

This subsection is compliant with [DVB-GEM 1.0.2] Section: 10. Application Signaling and extends [DVB-MHP 1.0.3] Section: 10.5.1 Encoding.
OCAP complies with Section 10.5.1 of [DVB-MHP 1.0.3] except where stated below:

The same application identifier MAY appear more than once in the set of AIT subtables. Where multiple entries of the same application are signaled, at most one SHALL be entered into the applications database. The implementation SHALL make this determination using the application addressing, priority, versioning, and launch order attributes that are available in the signaling.

11.2.2 Extensions to DVB-GEM (Normative)

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

The following information is normative for OCAP. It is specific to OCAP and is not contained within the [DVB-MHP 1.0.3].

11.2.2.1 Signaling of Unbound Applications

OCAP 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 affects 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 application signaling via the OCAP 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 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-MHP 1.0.3]. The maximum cycle time of the XAIT SHOULD be 10 seconds. 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 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-MHP 1.0.3] Syntax of the AIT, with the deviation specified by Section 11.2.1.3 and Section 11.2.1.5. OCAP provides the capability for an implementation to save a signaled XAIT to persistent storage. This XAIT (subsequently referred to as saved, stored, or persisted) can be used by the implementation to establish the OCAP environment when no XAIT is signaled on
the cable network. Section 20.2.1.2 describes implementation behavior when an XAIT is not available from the network.

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 an 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, unbound 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>
<thead>
<tr>
<th>application_id values</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000…0x3fff</td>
<td>Application_ids for unsigned applications</td>
</tr>
<tr>
<td>0x4000…0x5fff</td>
<td>Application_ids for signed applications without monitor permission</td>
</tr>
<tr>
<td>0x6000…0x7fff</td>
<td>Application_ids for dual signed unbound applications that use monitor permission</td>
</tr>
<tr>
<td>0x8000…0xffffd</td>
<td>Reserved for future use by DVB</td>
</tr>
<tr>
<td>0xfffe…0xffff</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

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-MHP 1.0.3] Data Errors.

11.2.2.3.3 Application Descriptor

An application descriptor as specified in Section 10.7.3 of the [DVB-MHP 1.0.3] SHALL be included in the XAIT for each application signaled. OCAP 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-MHP 1.0.3].
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 10.7.4.2 of the [DVB-MHP 1.0.3].

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 10.7.4.2 of the [DVB-MHP 1.0.3], and, as deviated by in OCAP 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-MHP 1.0.3], and as deviated by in OCAP 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 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 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 Section 11.2.1.7.

11.2.2.3.10 **Pre-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-MHP 1.0.3].

11.2.2.3.11 **DII 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-MHP 1.0.3].

11.2.2.3.12 **OCAP-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-MHP 1.0.3], and as deviated by in OCAP Section 11.2.1.1, all occurrences of DVB-J SHALL be recognized as an OCAP-J application type in an OCAP environment.

11.2.2.3.13 **OCAP-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-MHP 1.0.3], and as deviated by in OCAP Section 11.2.1.1, all occurrences of DVB-J SHALL be recognized as an OCAP-J application type in an OCAP environment.
11.2.2.3.14 **Abstract Service Descriptor**

OCAP extends [DVB-MHP 1.0.3] 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

<table>
<thead>
<tr>
<th></th>
<th>No. of Bits</th>
<th>Identifier</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract_service_descriptor() {</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>descriptor_tag</td>
<td>8</td>
<td>uimsbf</td>
<td>0x66</td>
</tr>
<tr>
<td>descriptor_length</td>
<td>8</td>
<td>uimsbf</td>
<td></td>
</tr>
<tr>
<td>service_id</td>
<td>24</td>
<td>uimsbf</td>
<td></td>
</tr>
<tr>
<td>reserved_for_future_use</td>
<td>7</td>
<td>uimsbf</td>
<td></td>
</tr>
<tr>
<td>auto_select</td>
<td>1</td>
<td>bslbf</td>
<td></td>
</tr>
<tr>
<td>for (i=0; i&lt;N; i++) {</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service_name_byte</td>
<td>8</td>
<td>uimsbf</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>service_id [24 bit]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract service defined by MSO</td>
</tr>
<tr>
<td>Abstract service defined by manufacturer</td>
</tr>
</tbody>
</table>

**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.

11.2.2.3.15 **Unbound Application Descriptor**

OCAP extends [DVB-MHP 1.0.3] 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

<table>
<thead>
<tr>
<th>No. of Bits Identifier Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>descriptor_tag              8 uimsbf 0x67</td>
</tr>
<tr>
<td>descriptor_length           8 uimsbf</td>
</tr>
<tr>
<td>service_id                  24 uimsbf</td>
</tr>
<tr>
<td>version_number              32 uimsbf</td>
</tr>
</tbody>
</table>

**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.

11.2.2.3.16 Privileged Certificate Descriptor

OCAP extends [DVB-MHP 1.0.3] 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

<table>
<thead>
<tr>
<th>No. of Bits Identifier Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>descriptor_tag              8 uimsbf 0x68</td>
</tr>
<tr>
<td>descriptor_length           8 uimsbf</td>
</tr>
<tr>
<td>for (i=0; i&lt;N; i++) {</td>
</tr>
<tr>
<td>for (j=0; j&lt;20; j++) {</td>
</tr>
<tr>
<td>certificate_identifier_byte 8 uimsbf SHA-1 Hash</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

**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.

11.2.2.3.17 Application Storage Descriptor

OCAP extends [DVB-MHP 1.0.3] 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

<table>
<thead>
<tr>
<th>No. of Bits</th>
<th>Identifier</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>descriptor_tag</td>
<td>8</td>
<td>uimsbf</td>
</tr>
<tr>
<td>descriptor_length</td>
<td>8</td>
<td>uimsbf</td>
</tr>
<tr>
<td>storage_priority</td>
<td>16</td>
<td>uimsbf</td>
</tr>
<tr>
<td>launch_order</td>
<td>8</td>
<td>uimsbf</td>
</tr>
</tbody>
</table>

`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.

11.2.2.3.18 Application mode descriptor

OCAP extends [DVB-MHP 1.0.3] 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. Applications that are signaled without this descriptor SHALL enter the default or legacy background mode, per Section 10.2.2.7.5, when their home environment transitions to the background state (see Section 10.2.2.4).

Table 11–11 - Application Mode Descriptor

<table>
<thead>
<tr>
<th>No. of Bits</th>
<th>Identifier</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>descriptor_tag</td>
<td>8</td>
<td>uimsbf</td>
</tr>
<tr>
<td>descriptor_length</td>
<td>8</td>
<td>uimsbf</td>
</tr>
<tr>
<td>mode</td>
<td>3</td>
<td>uimsbf</td>
</tr>
<tr>
<td>reserved</td>
<td>5</td>
<td>uimsbf</td>
</tr>
</tbody>
</table>

`descriptor_tag`: This 8 bit integer with a value 0x6F identifies this descriptor.

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

`mode`: Identifies the modes in which an application can run. The values of the mode variable are as follows:

Table 11–12 - Values for Mode variable of Application Mode Descriptor

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Default mode for applications that are not signaled or are otherwise environment state unaware, i.e., the legacy background mode. See Section 10.2.2.7.5.</td>
</tr>
</tbody>
</table>
1 Indicates that the application can only run in normal mode. Such applications SHALL be terminated when their home environment leaves the selected or presenting states.

2 Indicates that the application can run in cross_environment mode. This value is only valid in the XAIT. Any instance in the AIT SHALL be interpreted as 1, indicating that the application can only run in normal mode.

3 Indicates that the application can run in background mode.

4 Indicates that the application can run in paused mode, per section 10.2.2.7.4.

5 – 7 Reserved

### 11.2.2.4 Registered 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 11–13 - Registered API Descriptor

<table>
<thead>
<tr>
<th>No. of Bits</th>
<th>Identifier</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>uimsbf</td>
<td>0x6A</td>
</tr>
</tbody>
</table>

**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.

### 11.2.2.5 Addressable AIT and XAIT

Both the AIT and XAIT MAY contain signaling that associates a set of applications with a set of Host devices. This signaling is based on three descriptors: the addressing_descriptor, the addressable_application_descriptor, and the attribute_mapping_descriptor. For the following discussion, refer to Figure 11–1 and Figure 11–2. The addressing_descriptor provides a comparison mechanism that evaluates true or false for each Host device. The addressable_application_descriptor is linked to one or more addressing_descriptor(s). When an implementation "matches" multiple addressing_descriptor(s) in the same group, the
implementation SHALL choose the addressing_descriptor with the highest priority. Priority of an addressing_descriptor is determined by its priority field. If an implementation matches multiple addressing_descriptor(s) in the same group and with the same priority, the implementation MAY use any one of them. The implementation SHALL accept signaling for any applications with an addressable_application_descriptor that references (i.e., contains equal address_label fields) as a "matched" addressing_descriptor with the highest group priority identified by the implementation. This allows an addressable_application_descriptor to be used by the implementation based on one addressing_descriptor in one or more groups. An addressable_application_descriptor SHALL refer to one or more addressing_descriptor(s) that are defined in any section of an XAIT or AIT. The attribute_mapping_descriptor maps numerical identifiers used in the addressing_descriptor to standardized or network assigned string names.

Multiple attribute mapping descriptors can appear in the outer descriptor loop of any AIT or XAIT section

An application to be signaled with addressing will have a single addressable_application_descriptor

Multiple applications may appear in an AIT or XAIT

An addressable application descriptor may reference one or more addressing descriptors in its address_labels field

The attribute name in an attribute mapping descriptor must be the name of a system property or a name resolved through the Host Addressable Properties CableCARD resource

Multiple addressing descriptors may appear in the outer descriptor loop of any AIT or XAIT section

Addressing descriptors with the same group identifier belong to the same group. Applications associated with any of the highest priority matching addressing descriptors in each group are signaled

The attribute_id in an addressing comparison operation must match the attribute id in one of the attribute mapping descriptors

An addressing descriptor may have zero or more attribute comparison operations

Multiple addressing descriptors may appear in the outer descriptor loop of any AIT or XAIT section

Addressing descriptors with the same group identifier belong to the same group. Applications associated with any of the highest priority matching addressing descriptors in each group are signaled

An addressable application descriptor may reference one or more addressing descriptors in its address_labels field

The attribute name in an attribute mapping descriptor must be the name of a system property or a name resolved through the Host Addressable Properties CableCARD resource

Multiple applications may appear in an AIT or XAIT

An application to be signaled with addressing will have a single addressable_application_descriptor

Multiple attribute mapping descriptors can appear in the outer descriptor loop of any AIT or XAIT section

An addressable application descriptor may reference one or more addressing descriptors in its address_labels field

Figure 11–1 - Addressing Descriptors Relationship
An XAIT with these addressing elements will signal apps to various hosts as follows:

<table>
<thead>
<tr>
<th>Host Attributes</th>
<th>Applications that will be signaled</th>
</tr>
</thead>
<tbody>
<tr>
<td>ocap.cablecard.id = 1,</td>
<td>6002, 6005, 6006</td>
</tr>
<tr>
<td>ocap.cablecard.id = 2,</td>
<td>6002, 6005, 6006</td>
</tr>
<tr>
<td>ocap.cablecard.id = 4,</td>
<td>6001, 6004</td>
</tr>
<tr>
<td>ocap.cablecard.id = 4, ocap.dvrext.version = null</td>
<td>6001, 6005, 6006</td>
</tr>
<tr>
<td>ocap.cablecard.id = 4, ocap.dvrext.version = 2.0</td>
<td>6001, 6005, 6006</td>
</tr>
</tbody>
</table>

Figure 11–2 - Addressing Descriptors Relationship Example

11.2.2.5.1 Addressing Descriptor

The addressing_descriptor identifies a set of Host devices to which specific applications may be targeted. Zero or more addressing_descriptor(s) may appear in the common descriptor loop of any section in an XAIT or AIT. The address_expression_byte field SHALL contain one or more of the op_code based operations defined in this section. Each comparison operation is evaluated as true or false and placed in a logical stack. Each logical operation (e.g.,
AND, OR, NOT) causes one or two comparison results to be popped off the stack, evaluated for the logical
operation, and the result to be pushed on the stack. The final evaluation of the stack determines if the
addressing_descriptor is matched or not. When a stack is invalid and cannot be evaluated correctly, the
addressing_descriptor evaluates to false. A stack is invalid in the following circumstances:

- A logical operation requiring one comparison result is signaled, but the stack contains zero comparison
  results.
- A logical operation requiring two comparison results is signaled, but the stack contains one or zero
  comparison results.
- Multiple comparison results are on the stack, but no logical operation is signaled to evaluate them.
- Two comparison results being compared from different group identifiers.

The order of addressing_descriptor evaluation is the same order signaled in the common loop of an XAIT or AIT.
Other descriptor types MAY be signaled in the common loop intermingled with addressing_descriptor(s). All
sections in an AIT or XAIT must be processed before making final addressing_descriptor evaluations.

<table>
<thead>
<tr>
<th>Table 11–14 - addressing_descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Bits</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>descriptor_tag</td>
</tr>
<tr>
<td>descriptor_length</td>
</tr>
<tr>
<td>group_identifier</td>
</tr>
<tr>
<td>address_label</td>
</tr>
<tr>
<td>priority</td>
</tr>
<tr>
<td>for( i=0; i&lt;N; i++)</td>
</tr>
<tr>
<td>address_expression_byte</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

descriptor_tag: This 8-bit integer with value 0x6B identifies this descriptor.

descriptor_length: Number of bytes immediately following the length field.

group_identifier: An identifier that associates this addressing_descriptor with any other addressing_descriptors that
have the same group_identifier. For each distinct group_identifier, only the highest priority addressing_descriptor
that matches the host is used to select which applications are signaled.

address_label: The address_label is a numeric identifier for this addressing_descriptor. The address_label is
referenced by addressable_application_descriptor(s) associated with individual applications.

priority: The priority field is used to resolve addressing when a host matches more than one addressing_descriptor
in the same group.

address_expression_byte: The address_expression is a boolean expression based on various standard and
assignable attributes of the host device. It is expressed as a series of boolean stack operations. When this expression
evaluates to "true", the addressing_descriptor matches the host. This field MAY contain one or more of the op-code
based comparison structures.
11.2.2.5.1.1 Host Attribute Comparison Operation

The host_attribute_comparison operation compares the value of a Host device attribute with a constant in the instruction. A Host device attribute MAY be set by a privileged application using the org.ocap.application.AppManagerProxy.registerAddressingProperties method, MAY be queried from the security system, or MAY be set in a Java system property. If the comparison is true, then true is pushed onto the stack; otherwise, false is pushed onto the stack. If the attribute_id field does not match an attribute_id from an attribute_mapping_descriptor, the entire containing addressing_descriptor is discarded. When the attribute_id field matches an attribute_mapping_descriptor, the value associated with the property identified by the attribute_mapping_descriptor is compared to the string form of the attribute_value_char fields.

| host_attribute_comparison() { |
|------|----------|----------|
| op_code | 8 | uimsbf |
| attribute_id | 8 | uimsbf |
| security_attribute | 1 | bslbf |
| reserved_future_use | 7 | |
| attribute_value_length | 8 | uimsbf |
| for(i=0; i<N; i++) { |
| attribute_value_char | 8 | uimsbf |
|   } |

op_code: Determines the type of comparison to be done. Not all comparison types are legal for each attribute type; see legal op_code values in Table 11–16. If an illegal comparison type is signaled, the comparison SHALL evaluate to false.

<table>
<thead>
<tr>
<th>host_attribute_comparison operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Bits</td>
</tr>
<tr>
<td>host attribute less than</td>
</tr>
<tr>
<td>host attribute less than or equal</td>
</tr>
<tr>
<td>host attribute equal</td>
</tr>
<tr>
<td>host attribute greater than or equal</td>
</tr>
<tr>
<td>host attribute greater than</td>
</tr>
</tbody>
</table>

attribute_id: Identifies the attribute that is being compared. This value is matched to an attribute_id in the attribute_mapping_descriptor to determine which Java system property is compared.

security_attribute: When set to a value of 1, the implementation SHALL query the security system for a comparison value. For CableCARD query, see Host Addressable Property Query in [CCIF]. When set to a value of 0, the implementation SHALL determine key match using the address_name_char fields in the attribute_mapping_descriptor and compare to attributes set by the AppManagerProxy.registerAddressingProperties method and Java system properties.

attribute_value_length: Number of bytes in the attribute_value_char for loop.

attribute_value_char: String representation of the attribute value in UTF-8 format.

11.2.2.5.1.2 AND Logical Operation

The AND logical operation pops and examines the two top values on the stack. If both values that were on top of the stack are true, then true is pushed onto the stack; otherwise, false is pushed onto the stack.
Table 11–17 - AND operation

<table>
<thead>
<tr>
<th>No. of bits</th>
<th>Identifier</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>uimsbf</td>
<td>0x31</td>
</tr>
</tbody>
</table>

11.2.2.5.1.3 OR Logical Operation

The OR logical operation pops and examines the two top values on the stack. If either of the values that were on top of the stack were true, then true is pushed onto the stack; otherwise, false is pushed onto the stack.

Table 11–18 - OR operation

<table>
<thead>
<tr>
<th>No. of bits</th>
<th>Identifier</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>uimsbf</td>
<td>0x32</td>
</tr>
</tbody>
</table>

11.2.2.5.1.4 NOT Logical Operation

The NOT logical operation pops and examines top value on the stack. If the top value was true, then false is pushed onto the stack; otherwise, true is pushed on to the stack.

Table 11–19 - NOT operation

<table>
<thead>
<tr>
<th>No. of bits</th>
<th>Identifier</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>uimsbf</td>
<td>0x33</td>
</tr>
</tbody>
</table>

11.2.2.5.1.5 TRUE Logical Operation

The TRUE logical operation pushes a true value onto the stack. When used as the only operation in the lowest priority addressing descriptor in a group, this operation creates a default match for applications in the same group.

Table 11–20 - TRUE operation

<table>
<thead>
<tr>
<th>No. of bits</th>
<th>Identifier</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>uimsbf</td>
<td>0x34</td>
</tr>
</tbody>
</table>

11.2.2.5.2 Attribute Mapping Descriptor

The attribute_mapping_descriptor maps a string attribute name to a byte value. The host_attribute_comparison identifies an attribute with a byte value; see attribute_id value field in the host_attribute_comparison definition. This descriptor maps those values to corresponding string names that are Java system properties or addressable attributes properties. Addressable attributes are those that can be resolved through the host_properties_req() APDU of the Host Addressable Properties CableCARD resource or by privileged application via the AppManagerProxy class. If an attribute_id is mapped to more than one Java system property or addressable attribute, it is implementation-specific regarding which mapping is used. When taken as a string, if the attribute_name_char fields do not match a Java system property or a Property key passed to the AppManagerProxy.registerAddressingProperties method, and the attribute_name_char fields cannot be resolved to a property value using the host_properties_req() APDU, then the descriptor SHALL be discarded.
Table 11–21 - addressing_attribute_descriptor

<table>
<thead>
<tr>
<th>No. of Bits</th>
<th>Identifier</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute_mining_descriptor() {</td>
<td></td>
<td></td>
</tr>
<tr>
<td>descriptor_tag 8 uimsbf 0x6C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>descriptor_length 8 uimsbf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>attribute_id 8 uimsbf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>for( i=0; i&lt;N; i++) {</td>
<td></td>
<td></td>
</tr>
<tr>
<td>attribute_name_char 8 uimsbf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**descriptor_tag:** The tag value for this descriptor is 0x6C.

**descriptor_length:** The length of this descriptor in bytes.

**attribute_id:** Identifier that corresponds with the attribute_name_char field.

**attribute_name_char:** The name of the attribute encoded in UTF-8 format.

11.2.2.5.3 Addressable Application Descriptor

The addressable_application_descriptor associates an application with one or more addressing_descriptor(s) and is used to signal an application for a specific set of Host devices. For this purpose, the addressable_application_descriptor is used instead of the application_descriptor defined in [DVB-MHP 1.1.3]. The addressable_application_descriptor extends the application_descriptor with a list of one or more address_label fields that associate the application with specific addresses signaled in corresponding addressing_descriptor address_label fields. For all other specification, the addressable_application_descriptor SHALL be processed the same as an application_descriptor.

Table 11–22 - addressable_application_descriptor

<table>
<thead>
<tr>
<th>No. of Bits</th>
<th>Identifier</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>addressable_application_descriptor() {</td>
<td></td>
<td></td>
</tr>
<tr>
<td>descriptor_tag 8 uimsbf 0x6D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>descriptor_length 8 uimsbf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>application_profiles_length 8 uimsbf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>for( i=0; i&lt;N; i++) {</td>
<td></td>
<td></td>
</tr>
<tr>
<td>application_profile 16 uimsbf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>version.major 8 uimsbf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>version.minor 8 uimsbf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>version.micro 8 uimsbf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>service_bound_flag 1 bslbf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>visibility 2 bslbf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>reserved_for_future_use 5 bslbf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>application_priority 8 uimsbf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>address_labels_length 8 uimsbf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>for( i=0; i&lt;n; i++) {</td>
<td></td>
<td></td>
</tr>
<tr>
<td>address_label 8 uimsbf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>for( i=0; i&lt;N; i++) {</td>
<td></td>
<td></td>
</tr>
<tr>
<td>transport_protocol_label 8 uimsbf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**descriptor_tag:** This 8-bit integer with value 0x6D identifies this descriptor.
descriptor_length: Number of bytes immediately following the length field.

**application_profiles_length, application_profile, version.major, version.minor, version.micro, service_bound_flag, visibility, application_priority, transport_protocol_label:** See [DVB-MHP 1.1.3] section 10.7.3 Application Descriptor.

address_labels_length: Length of the address_label loop in bytes.

address_label: References the address_label in an addressing_descriptor.

### 11.2.2.5.4 Backwards Compatibility

In order to make addressable_application_descriptor signaling compatible with application_descriptor signaling, the following rules are asserted:

- When no addressable_application_descriptor(s) are signaled in an AIT or XAIT, any application_descriptor(s) that are present SHALL be processed without consideration of application addressing, even if addressing_descriptor(s) are present.
- When addressing_descriptor(s) are present in an AIT or XAIT and none of them evaluate to true, any application_descriptor(s) that are present SHALL be processed without consideration of application addressing.
- When an addressing_descriptor is present in an AIT or XAIT and evaluates to true and at least one addressable_application_descriptor references it, the implementation SHALL NOT process any application_descriptor that MAY be present in the same AIT or XAIT.
- An addressing_descriptor and addressable_application_descriptor have AIT or XAIT scope. When an addressable_application_descriptor was signaled with an addressing_descriptor that evaluated true but in a subsequent table evaluates false, the addressable_application_descriptor SHALL be treated as if it did not appear in signaling, i.e., the application SHALL be destroyed if running and removed from any AppsDatabase.

### 11.2.2.5.4.1 Mandatory Descriptors (informative)

Based on assertions made in [DVB-MHP 1.1.3] section 10.4.1 regarding mandatory descriptors, it is expected that OCAP devices will ignore an entire application-specific entry in an AIT or XAIT when an application_descriptor or addressable_application_descriptor is not present in the application's descriptor loop. Consequently, an AIT or XAIT application entry based on an addressable_application_descriptor is discarded by OCAP Host devices that do not implement a version of OCAP that requires compliance with the addressable_application_descriptor.

### 11.2.2.5.5 Host Addressable Attribute Management

The following rules are given for management of addressable attributes:

- When a privileged application registers an addressable attribute value by using the org.ocap.application.AppManagerProxy API the implementation SHALL re-evaluate the most recently received XAIT and AIT if it contains addressable attributes.
- Once a Host addressable property is retrieved from the security system, the implementation SHALL poll the value every 60 seconds by sending a new Host addressable property message to the security system using the same key.
  
  a. When a new XAIT or AIT drops an addressable attribute, the implementation SHALL stop polling it.
  
  b. If subsequent retrieval of a value changes, the implementation SHALL re-evaluate the most recently received XAIT and AIT that contains addressable attributes.
• When evaluating an XAIT or AIT that contains addressable attributes, any application entry that is evaluated as false is treated as if it were not part of the table.
12 APPLICATION STORAGE

This section contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

This section describes the management of applications that are stored on the OCAP 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

Section 12.2.7, Authentication of Stored Applications of OCAP, corresponds to section 12.6.1 of [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2] as follows:

| Table 12–1 - Correlation between OCAP and [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3] |
|---------------------------------|---------------------------|-----------------|-----------------|
| OCAP                           | [DVB-GEM 1.0.2] Section   | GEM Compliance  | [DVB-MHP 1.0.3] Section | MHP Compliance |
| 12.2.7 Authentication of Stored Applications | No Corresponding Section | OCAP Specific Extension | 12.6.1 General Principles | Compliance |

12.2 OCAP Requirements

12.2.1 Introduction

The loading and launching of an OCAP application may benefit from the application being stored on the terminal. An OCAP 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 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 terminal are sourced from the following set of different sources:

- The terminal manufacturer by initial installation or common download,
- The MSO through signaling in the XAIT,
- The MSO through other sources managed in the Monitor Application.

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

12.2.2 Storing Host device 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]. 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>
<thead>
<tr>
<th>storage_priority value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>This application SHALL NOT be stored</td>
</tr>
<tr>
<td>1-10</td>
<td>Reserved for future use</td>
</tr>
<tr>
<td>11-255</td>
<td>This application is stored according to the specified priority value and subject to available storage constraints.</td>
</tr>
</tbody>
</table>

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. As soon as the implementation determines from signaling that an application has sufficient storage priority to be stored, it SHALL download and store the application, whether or not the application is to be launched at that time. 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.

If an application download is interrupted due to any file download failure and the protocol_id is 0x101, then upon receipt of the same table in which the application was delivered, i.e., AIT or XAIT with matching version number, the implementation SHALL re-evaluate the application storage_priority, and if the application storage needs to be completed, SHALL re-commence download of the application. Also see partial storage rules in Section 12.2.9.

### 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 OcCapAppAttributes.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 
AppComponentProxy.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 Services Database (Informative)

The presence of applications in storage does not imply entry into the services database. Entry into the services database (and subsequently the applications database) is as defined in Section 10. More specifically, following a reboot or full power-cycle, stored applications are not entered into the services and applications databases unless they are signaled or registered during or after the boot process.

12.2.6 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.

When an XAIT is not accessible via OOB (e.g., no CableCARD is inserted, connected to a basic OpenCable network etc.), it is not required to remove applications.

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.
NOTE: As with applications loaded from the network, authentication failures in stored applications do not prevent the OCAP implementation from attempting to launch the Xlet. [DVB-MHP 1.0.3] section 12.6.1 describes how to treat files that did not authenticate, and Section 14.2.1.7 of this specification adds an extension.

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 an OCAP-J application as defined in the application signaling. By convention, the name of this file is:

'o cap. storage. ooo oooo o. aaaa'

where:

ooo oooo 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".

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 unescaped 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.

### 12.2.9 Partially-Stored Applications

During application download and storage, some event may occur, e.g., outage, that causes the application download to fail while in progress and after some of the application files have been stored. In such a case, the application will be partially stored. If an ADF is present, the implementation SHALL retain the partially-stored application based on the storage priority it was signaled with. The implementation SHALL NOT launch a partially stored application. If further attempts are made to download a partially-stored application, then the implementation SHALL NOT attempt to download files that have already been downloaded. The implementation SHALL use the ADF to determine the files that still need to be downloaded in order to complete application download.
13 EXECUTION ENGINE PLATFORM

This section contains requirements that are extensions to [DVB-GEM 1.0.2] 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 1.0.2] 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 1.0.2] Section. Section 13 (this section) of OCAP corresponds to Section 11 of [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2] as follows:

| Table 13–1 - Correlation between OCAP and [DVB-GEM 1.0.2] |
|----------------------------------|-----------------|-----------------|
| OCAP Section | [DVB-GEM 1.0.2] 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 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 OCAP | 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 1.0.2] Section.

This section corresponds to [DVB-GEM 1.0.2] clause 11.

As a consequence of Section 6.1, the Execution Engine is as specified in [DVB-GEM 1.0.2] Clause 11, "DVB-J Platform".

13.2.1 Methods working on many locator types

This subsection complies with [DVB-GEM 1.0.2] Section 11.11.11. When the `javax.tv.locator.LocatorFactory.transformLocator` method is called and the parameter is a source_id based locator, and if the frequency and program_number mappings are known for the source_id, the implementation SHALL return a locator with those terms. If the frequency or program_number to source_id mappings are not known, the implementation SHALL return the parameter locator.

13.3 OCAP Extensions to GEM

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.
13.3.1 General Issues

13.3.1.1 Basic Considerations

OCAP applications SHALL NOT use additional public or protected methods or fields in the \texttt{org.ocap.}\* namespace that are not listed in this specification or a published extension to this specification. An extension complies with Section 13.3.14.2 of this specification with a system property 'ocap.api.option.xxx' defined. Applications that are designed to scale on multiple revisions of this specification SHALL only use \texttt{org.ocap.}\* methods or fields appropriate to the version of the platform on which they are running.

OCAP applications SHALL NOT define classes or interfaces in any package namespace defined in this specification. OCAP terminals SHALL enforce this using the \texttt{SecurityManager.checkPackageDefinition} mechanism.

NOTE: (informative) This is consistent with the rules for packages defined by MHP, as specified in [DVB-MHP 1.0.3] clause 11.2.1 as modified in [DVB-GEM 1.0.2] clause 11.2.

13.3.1.2 Class Loading Shared Classes

This section extends and modifies [DVB-GEM 1.0.2] clause 11.2, specifically [DVB-MHP 1.0.3] clause 11.2.3 Class Loading, and provides a mechanism for sharing class files between applications; see Section 21.2.1.21.

As specified by [DVB-MHP 1.0.3] 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 \texttt{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 \texttt{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 \texttt{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 \texttt{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-MHP 1.0.3] clause 11.2.5 which is included in [DVB-GEM 1.0.2] clause 11.2.

13.3.1.4 Event Model in OCAP APIs

Each class in \texttt{org.ocap.}\* inheriting from \texttt{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-MHP 1.0.3] clause 11.2.7 which is included in [DVB-GEM 1.0.2] clause 11.2.

### 13.3.1.5 Tuning as a Side-Effect

No OCAP API SHALL cause tuning unless explicitly specified as such.

**NOTE:** (informative) This is consistent with the rule for MHP APIs, as specified in [DVB-MHP 1.0.3] clause 11.2.8 which is included in [DVB-GEM 1.0.2] clause 11.2.

### 13.3.1.6 Java Class Name and File Name Lengths

This section clarifies the requirements of MHP 1.0 section 14.6.

A fully conformant OCAP implementation SHALL support Java class names of at least 128 characters and fully qualified Java class names of at least 255 characters.

A fully conformant OCAP implementation SHALL support filename lengths, for files loaded from any source, of at least 255 characters and support file paths (i.e., the concatenation of the filename, directory separators, and directory names) of at least 1024 characters.

### 13.3.2 OCAP Platform APIs

OCAP requires the following APIs:

#### 13.3.2.1 `org.ocap` Package

OCAP terminals SHALL support the `org.ocap` package as defined in Annex O.

#### 13.3.2.2 `org.ocap.application` Package

OCAP terminals SHALL support the `org.ocap.application` package as defined in Annex G.

#### 13.3.2.3 `org.ocap.event`

OCAP 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-MHP 1.0.3] clause 11.3.2.2, which is included in [DVB-GEM 1.0.2] 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 the `org.ocap.event.EventManager.getInstance()` method SHALL return the object for the `org.ocap.event.EventManager` class.

OCAP terminals SHALL report instances of `org.ocap.ui.event.OCRcEvent` through the normal `java.awt.event` event mechanism. This is facilitated by the inheritance from `java.awt.event.KeyEvent`. 
13.3.2.4 org.ocap.net

OCAP terminals SHALL support the org.ocap.net package as defined in Annex I.

OCAP terminals SHALL return an instance of org.ocap.net.OcapLocator or org.ocap.net.URLLocator from javax.tv.locator.LocatorFactory, as discussed in [DVB-GEM 1.0.2] clause 11.3.

OCAP terminals SHALL support the requirements of Annex B.

13.3.2.5 org.ocap.hardware

OCAP terminals SHALL support the org.ocap.hardware package as defined in Annex F.

13.3.2.6 org.ocap.hardware.pod

OCAP terminals SHALL support the org.ocap.hardware.pod package as defined in Annex R.

13.3.2.7 org.ocap.media

OCAP terminals SHALL support the org.ocap.media package as defined in Annex S.

13.3.2.8 org.ocap.resource

OCAP terminals SHALL support the org.ocap.resource package as defined in Annex L.

13.3.2.9 org.ocap.service

OCAP terminals SHALL support the org.ocap.service package as defined in Annex P.

13.3.2.10 org.ocap.system

OCAP terminals SHALL support the org.ocap.system package as defined in Annex Q.

13.3.2.11 org.ocap.ui and org.ocap.ui.event

OCAP terminals SHALL support the org.ocap.ui and org.ocap.ui.event packages, as defined in Annex E.

13.3.2.12 org.ocap.si

OCAP terminals SHALL support the org.ocap.si package as defined in Annex T.

13.3.2.13 org.ocap.environment

OCAP host devices SHALL support the org.ocap.environment package as defined in Annex Y. On OCAP host devices which only support a cable environment, the behavior of the org.ocap.environment package SHALL be as follows:

- `select` SHALL immediately post an `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 an 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 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 terminals SHALL support the requirements of Section 21.

13.3.5 Java Platform APIs

As a consequence of Section 6.1, OCAP extends [DVB-GEM 1.0.2], section 11.3 to include all packages defined in [JSR 217]. The following notes apply to the Java Platform APIs as defined in [DVB-MHP 1.0.3] Section 11.3.1.

[DVB-MHP 1.0.3] Section 11.3.1.1 "java.lang" contains bullet (g) that begins with, "Only the following properties are required to be supported for System.getProperty() and System.getProperties()". This SHALL be considered to be replaced with, "In addition to the system properties required by PBP 1.1 ([JSR 217] and [JSR 280]), the following properties are required to be supported for System.getProperty()".

[DVB-MHP 1.0.3] Section 11.3.1.1 "java.lang" contains bullets (a), (b), (d), (h), and (i), which are redundant in light of full PBP 1.1 support. These statements MAY be disregarded without affecting DVB-GEM compliance.

[DVB-MHP 1.0.3] Sections 11.3.1.2 "java.lang.reflect", 11.3.1.7 "java.beans", and 11.3.1.8 "java.math" are redundant in light of full PBP 1.1 support. These sections MAY be disregarded without affecting DVB-GEM compliance.

[DVB-MHP 1.0.3] Section 11.3.1.2 "java.util" contains the following statement: "The format used for the java.util.Properties.save() and java.util.Properties.load() methods shall be that specified for those methods in JDK 1.2.2", which is redundant in light of full PBP 1.1 support. This statement MAY be disregarded without affecting DVB-GEM compliance.

[DVB-MHP 1.0.3] Section 11.3.1.4 "java.util.zip" includes a subset of the java.util.zip package as defined in [JSR 217]. This section MAY be disregarded without affecting DVB-GEM compliance.

[DVB-MHP 1.0.3] Section 11.3.1.5 "java.io" specifies constraints on the java.io package. These constraints SHALL additionally be considered to apply to the javax.microedition.io package as defined in [JSR 217].

13.3.6 Presentation APIs

The Presentation APIs are those interfaces which support the presentation of mixed media and graphical user interfaces in an OCAP application.

13.3.6.1 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`, `java.awt.event`, and `java.awt.image`, as specified by [DVB-MHP 1.0.3] clause 11.4.1, which is included in [DVB-GEM 1.0.2] clause 11.4. As a consequence of Section 6.1, OCAP extends [DVB-GEM 1.0.2] and requires support of the entire `java.awt` package, as well as the `java.awt.color`, `java.awt.event`, `java.awt.font`, `java.awt.image`, and `java.awt.im` sub-packages as defined in [JSR 217].
- Parts of `javax.microedition.xlet` as a consequence of Section 6.1 and as specified by [JSR 217].
- `org.havi.ui` and `org.havi.ui.event`, as specified by [DVB-MHP 1.0.3] clause 11.4.1.2, which is included in [DVB-GEM 1.0.2] clause 11.4.
- Parts of various sub-packages of `javax.tv`, as specified by [DVB-MHP 1.0.3] clause 11.4.1.2, which is included in [DVB-GEM 1.0.2] clause 11.4.
- `org.dvb.ui`, as specified by [DVB-MHP 1.0.3] clause 11.4.1.3, which is included in [DVB-GEM 1.0.2] clause 11.4.
- `org.ocap.ui`, as specified in this specification; see Annex D.

### 13.3.6.1.1 Extensions (normative)

As a consequence of Section 6.1, [DVB-GEM 1.0.2] clause 11.4 is extended to integrate graphical user interface APIs defined in [JSR 217] as follows:

The "single instance of an application-created Frame" that "is permitted per GraphicsDevice" will be created by the OCAP implementation for the default `HGraphicsDevice`. Attempts by OCAP applications to construct instances of Frame for the default `HGraphicsDevice` SHALL cause the constructor to throw `java.lang.UnsupportedOperationException`.

The methods `javax.microedition.xlet.XletContext.getContainer` and `javax.tv.graphics.TVContainer.getRootContainer` SHALL return the same as would be returned by a call to `org.havi.ui.HSceneFactory.getDefaultHScene()` under the same circumstances.

Any changes made to the `org.havi.ui.HGraphicsConfiguration` of an `org.havi.ui.HGraphicsDevice` SHALL be reported via the `java.awt.GraphicsDevice` and `java.awt.GraphicsConfiguration` APIs, as appropriate. For each `HGraphicsConfiguration`, there SHALL be a `GraphicsConfiguration` where `getBounds()` returns the same values as `HGraphicsConfiguration.get PixelResolution()`.

NOTE: Application developers SHOULD NOT rely on OCAP devices supporting full-screen exclusive mode; thus, as permitted by [JSR 217], the method `java.awt.GraphicsDevice.isFullScreenSupported` MAY always return false.

### 13.3.6.2 Handling of Input Events

This subsection extends and modifies [DVB-MHP 1.0.3] Section 11.4.1.4.

**NOTE:** (informative) The OCAP user input event package, `org.ocap.event`, extends the `org.dvb.event` package. This is specified in Section 13.3.2.3.

[DVB-MHP 1.0.3] defines the concept of a "resident navigator", which is incorporated into [DVB-GEM 1.0.2]. OCAP does not directly support this concept; instead, an OCAP terminal supports baseline "Watch TV" functionality (see Section 20) and supports OCAP applications that embody the functionality of the "resident navigator". OCAP 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 applications.
and MAY consume input events without passing them on to other OCAP 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 1.0.2].

13.3.6.2.1 Application recommendations for VK_TELETEXT (informative)

OCAP does not require support for the VK_TELETEXT input event. Thus, some of the application recommendations in [DVB-MHP 1.0.3] clause 11.4.1.4 may not be applicable. OCAP applications should not rely on the VK_TELETEXT key.

13.3.7 Streamed Media API

13.3.7.1 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 [Java TV]. The following extended APIs are also supported:

- org.davic.media, as specified by [DVB-MHP 1.0.3] clause 11.4.2.5.2 as included in and modified by [DVB-GEM 1.0.2] clause 11.4.
- org.dvb.media, as specified by [DVB-MHP 1.0.3] clause 11.4.2.5.1 as included in and modified by [DVB-GEM 1.0.2] clause 11.4.

The language pertaining to DVBLocator in [DVB-MHP 1.0.3] clause 11.4.2.2 is modified by [DVB-GEM 1.0.2] clause 11.4; hence, these modifications apply to OCAP. Specifically, Section 13.3.2.4 introduces OCAPLocator, which takes the place of DVBLocator.

Classes related to subtitling required in [DVB-MHP 1.0.3] clause 11.4.2.5.1 are made optional by [DVB-GEM 1.0.2] clause 11.4; hence, they are optional in OCAP.

13.3.7.2 OCAP Extensions to the Framework

13.3.7.2.1 Additional constraints on Streamed Media API Extensions

In addition to the requirements of [DVB-GEM 1.0.2], OCAP 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-MHP 1.0.3] clause 11.4.2.5, as included in and modified by [DVB-GEM 1.0.2], clause 11.4. These modifications include removing the requirement for classes related to subtitling and the DVB CA system.

In response to calls to javax.media.Controller.getControl() and javax.media.Controller.getControls(), OCAP implementations SHALL return instances of org.dvb.media.VideoFormatControl that are also instances of org.ocap.media.VideoFormatControl.
Controls that are returned by `Player.getControls()` need not have an associated GUI component; therefore, calls to `Control.getControlComponent()` may return null.

Developers of MHP applications should not rely on the presence of the following classes or interfaces:

- `javax.media.CachingControl` (unless returned by a call to a JMF method)
- `javax.media.CachingControlEvent`

An MHP implementation need not return a `CachingControl` object from a call to `Player.getControls()`; however, this is not prohibited. If the specified `Player` is capable of playing audio, the return value of `Player.getControls()` SHALL include a `GainControl` object; likewise, if the specified `Player` is capable of playing audio, `Player.getGainControl()` SHALL return a `GainControl`. However, parameters passed to `GainControl.setLevel(float)` and `GainControl.setDB(float)` MAY be ignored such that the audio volume is unchanged. The volume that is set using the `GainControl` interface SHALL NOT be greater than the system volume level, i.e., the gain control may only change the volume between mute and the volume level at the time the application was started. This system volume level shall be represented as 1.0.

**NOTE:** Calls to `GainControl.setMute()` must always be respected.

### 13.3.8 Data Access APIs

**NOTE:** (informative) Data Access APIs are specified in [DVB-MHP 1.0.3] clause 11.5, as included in and modified by [DVB-GEM 1.0.2], clause 11.5.

OCAP extends [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3] 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.

#### 13.3.8.1 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.io`, as specified by [DVB-MHP 1.0.3] Section 11.3.1.5, "java.io", which is included in [DVB-GEM 1.0.2] clause 11.3. As a consequence of Section 6.1, OCAP extends [DVB-GEM 1.0.2] and additionally requires support for the `javax.microedition.io` package as defined in [JSR 217].
• java.net, as specified by [DVB-MHP 1.0.3] Section 11.3.1.6 "java.net" which is included in [DVB-GEM 1.0.2] clause 11.3. As a consequence of Section 6.1, OCAP extends [DVB-GEM 1.0.2] and requires support of the entire java.net package as defined in [JSR 217].

• javax.tv.net, as specified by [DVB-MHP 1.0.3] clause 11.5.2 which is included in [DVB-GEM 1.0.2] clause 11.5.

• org.dvb.dsmcc, as specified by [DVB-MHP 1.0.3] clause 11.5.1 as included in and modified by [DVB-GEM 1.0.2] clause 11.5, with the further semantics required by [DVB-MHP 1.0.3] annex P.

• org.dvb.io.persistent, as specified by [DVB-MHP 1.0.3] as included in and modified by [DVB-GEM 1.0.2].

13.3.8.2 Broadcast Transport Protocol Access API

The OCAP implementation SHALL differentiate the base directory for each OCAP application. When an application is signaled via the XAIT or AIT, the OCAP 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 dvb.j.location.base and the AppAttributes.getProperty() method. This behavior is required by [DVB-GEM 1.0.2] 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 DSMCCObject 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.

13.3.8.3 Support for IP over the Return Channel (informative)

Support for IP over the return channel is as specified in [DVB-MHP 1.0.3] clause 11.5.3 as included in [DVB-GEM 1.0.2], clause 11.5. See also the notes in Section 13.5.4 of this specification.

OCAP supports org.ocap.net package, as specified in Section 13.3.2.4. See also Section 13.3.15.1.

13.3.8.4 MPEG-2 Section Filter API

NOTE: (informative) Portions of this API are specified by [DVB -MHP 1.0.3] clause 11.5.4, as included in [DVB-GEM 1.0.2] clause 11.5. As required by MHP, the errata in [DVB-MHP 1.0.3] 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]". 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 16.2.1.7. 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], it specifies an instance of org.ocap.mpeg.PODExternalChannel to the org.davic.mpeg.sections.SectionFilterGroup.attach() method. The org.ocap.mpeg.PODExternalChannel 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] 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].

13.3.8.5 Persistent Storage API

This section extends [DVB-MHP 1.0.3] clause 11, which is included by [DVB-GEM 1.0.2] 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.

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 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 OCAP 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 the org.ocap.storage.ExtendedFileAccessPermissions 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 `<device designator>"/OCAP_LSV"<orgId>/<appId>/<logical storage volume name>`. The device designator is implementation-specific and SHALL be unique to the storage device. The logical storage volume name is the name assigned by an application when the volume was allocated. The "/OCAP_LSV/" directory SHALL be reserved for `LogicalStorageVolume` allocation and SHALL always be placed off a root directory and follow a device designator. Relative file paths MAY be appended to a logical storage volume path using the "/" separator. Applications will be granted read-only access to the "/OCAP_LSV/" 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. The access set by the `StorageProxy.allocateLogicalStorageVolume` applies only to the directory represented by the logical storage volume name. It is implementation-specific regarding whether the <orgId> or <appId> portions of the logical storage volume path are given write permission. If write permission is given and files are written to those directories, they MAY be lost when the logical storage volume is re-created as defined in Section 13.3.8.5.1.

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.

13.3.8.5.1 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 from the root file separator are identical; see Section 13.3.8.5 for path format definition. The `<device_designator>` term is implementation-specific. 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 containing the substring "/OCAP_LSV/"<orgId>/<appId>/<name>, where orgId is an organization identifier and appId is an application Id as defined in [DVB-MHP 1.0.3], 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-MHP 1.0.3], 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.

13.3.8.5.2 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 DetachableStorageOption 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.

13.3.8.5.3 Removable Devices

Any StorageProxy that represents a storage device with a removable storage media bay that supports the insertion or removal of storage media while power is applied SHALL contain an instance of RemovableStorageOption. A StorageProxy MAY be both detachable and support removable media (e.g., IEEE 1394-connected memory stick bay), in which case the StorageProxy SHALL contain distinct instances of RemovableStorageOption and DetachableStorageOption.

13.3.8.5.4 Implementation Provided LogicalStorageVolume

Whenever a detachable or removable storage device is connected to a Host device and the storage device is preformatted and contains data files that are not under the "/OCAP_LSV" 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.

**NOTE:** (informative) As a consequence, all files in an implementation provided LogicalStorageVolume are readable by all OCAP-J applications.
13.3.8.5.5  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.

13.3.9  Service Information and Selection APIs

These APIs are specified in [DVB-GEM 1.0.2] clause 11.6.

13.3.9.1  Uses (informative)

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.

13.3.9.2  DVB Service Context

All instances of ServiceContext returned by the OCAP terminal SHALL be instances of DvbServiceContext; see [DVB-MHP 1.1.3] Annex AK.

13.3.9.3  OCAP Extensions to the Framework

OCAP terminals SHALL NOT generate SelectionFailedEvents with reason codes of CONTENT_NOT_FOUND, MISSING_HANDLER, or TUNING_FAILED. In OCAP, such errors are not considered selection failures, as alternative content is to be presented instead. In a case where such error conditions would be encountered during service selection, regardless of previous ServiceContext state, alternative content SHALL be presented as prescribed for the CA_REFUSAL case in Section 16.2.1.7. Presentation of such alternative content SHALL be reported via generation of an org.ocap.service.AlternativeContentErrorEvent (see Annex P), which indicates the error reason. The implementation SHALL recognize any changes that may indicate that the error condition that led to alternative content presentation has been lifted and restore normal content if possible. The following table outlines the minimum conditions for reconsidering normal content presentation:

### Table 13–2 - Triggers for reconsideration of normal content presentation

<table>
<thead>
<tr>
<th>Alternative Content trigger</th>
<th>Reason Code</th>
<th>Normal Content trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parental control settings prevent presentation</td>
<td>RATING_PROBLEM</td>
<td>Change in content ratings or parental control settings eliminating the need to block</td>
</tr>
<tr>
<td>CA status prevents presentation</td>
<td>CA_REFUSAL</td>
<td>Change in CA status resulting in service entitlement</td>
</tr>
<tr>
<td>Content could not be found in the network</td>
<td>CONTENT_NOT_FOUND</td>
<td>Change in component information for the service enabling location of requested content</td>
</tr>
<tr>
<td>Handler necessary to present content is absent</td>
<td>MISSING_HANDLER</td>
<td>Change in component information for the service enabling use of a known handler</td>
</tr>
<tr>
<td>Lack of tuning information or errors encountered during tuning</td>
<td>TUNING_FAILURE</td>
<td>Change in tuning information for the service enabling tuning to the service</td>
</tr>
</tbody>
</table>
Following successful presentation of normal content, OCAP terminals SHALL NOT terminate service presentation due to the error conditions defined in Table 13–2. Instead, alternative content SHALL be presented as described for the initial service selection case until normal content presentation can be restored. Presentation of such alternative content SHALL be reported via generation of an `org.ocap.service.AlternativeContentErrorEvent` (see Annex P), which indicates the error reason. The implementation SHALL recognize any changes that may indicate that the error condition that led to alternative content presentation has been lifted and restore normal content if possible. Table 13–2 outlines the minimum conditions for reconsidering normal content presentation.

### 13.3.10 Common Infrastructure APIs (informative)

In addition to the requirements of [DVB-GEM 1.0.2] 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 1.0.2] clause 11.7 specifies APIs in the packages `javax.tv.xlet`, `java.rmi`, and `org.dvb.application`.

#### 13.3.10.1 APIs to support DVB-J application lifecycle (normative)

As a consequence of Section 6.1, OCAP extends [DVB-GEM 1.0.2] and includes the `javax.microedition.xlet` package as defined by [JSR 217]. The requirements of [DVB-GEM 1.0.2] clause 11.7.1 are additionally extended as follows:

The package `javax.microedition.xlet` contains classes similar to those defined in the `javax.tv.xlet` package by [Java TV]. In either case, the `XletContext` passed to the `Xlet` SHALL implement both `javax.microedition.xlet.XletContext` and `javax.tv.xlet.XletContext`.

The implementation SHALL determine which `xlet` interface is implemented by the initial class of an application, and use the corresponding application management API. In the case where the initial class of an OCAP application implements both `javax.tv.xlet.Xlet` and `javax.microedition.xlet.Xlet`, the implementation SHALL use the `javax.tv.xlet` application management API.

#### 13.3.10.2 Inter-Application Communication (normative)

As a consequence of Section 6.1, OCAP extends [DVB-GEM 1.0.2] and includes the entire `java.rmi` and `java.rmi.registry` packages and `javax.microedition.xlet` and `javax.microedition.xlet.ixc` packages as defined by [JSR 217]. The requirements of [DVB-GEM 1.0.2] clause 11.7.3 are additionally extended as follows:

The `org.dvb.io.ixc` registry SHALL be mapped to the `javax.microedition.xlet.ixc` registry as follows:

```
<table>
<thead>
<tr>
<th>DVB Registry Name</th>
<th>J2ME Registry Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>dvb:/service/service_context_id/signed/organization_id/application_id/name</td>
</tr>
<tr>
<td>name</td>
<td>dvb:/service/service_context_id/unsigned/organization_id/application_id/name</td>
</tr>
<tr>
<td>/organization_id/</td>
<td>dvb:/service/service_context_id/signed/organization_id/application_id/name</td>
</tr>
<tr>
<td>application_id/name</td>
<td>dvb:/service/service_context_id/unsigned/organization_id/application_id/name</td>
</tr>
</tbody>
</table>
```

Organization_id, application_id, and service_context_id are as defined for unsigned applications in 14.2.2.9.
A DSMCCObject, by virtue of being Serializable, may be passed to or returned from Remote object methods via remote copy; see [DVB-MHP 1.0.3] 11.7.3.1.2 Objects Passed by Remote Copy. For example, when an application attaches an org.dvb.dsmcc.ServiceDomain, it may bind objects to the IxcRegistry that have methods that can return an org.dvb.dsmcc.DSMCCObject. When a DSMCCObject is serialized and deserialized as part of IXC remote copy, the following behaviors apply:

- A DSMCCObject is serializable only due to extension of java.io.File. Attributes that are specific to DSMCCObject SHALL NOT be preserved through serialization.
- The implementation SHALL preserve the pathname of a serialized DSMCCObject. Definition of a relative pathname that has been deserialized is implementation-specific and may not provide access to the desired file. For this reason, an absolute pathname should be used by an application when it creates a DSMCCObject to be serialized.
- Upon completion of deserialization, the implementation SHALL set the deserialized DSMCCObject to the unloaded state.
- For purposes of a deserialized DSMCCObject, a ServiceDomain attachment and cache SHALL be visible across application virtual machines. The name space of a ServiceDomain SHALL remain constant across application virtual machines, and a file in an attached ServiceDomain can be accessed from any application virtual machine using a DSMCCObject created with the correct absolute pathname.
- When a file referenced by a deserialized DSMCCObject is present in cache, the file SHALL be accessible by the DSMCCObject given appropriate file permissions. In this case, a call to one of the DSMCCObject methods, including asynchronousLoad, synchronousLoad, and getSigners(boolean) SHALL NOT cause the file to be reloaded from a carousel or otherwise copied, and the DSMCCObject SHALL be transitioned to the loaded state.
- The state of a serialized DSMCCObject is separate from the state of cache as defined by the DSMCCObject description in [DVB-MHP 1.0.3] Annex P.

13.3.11 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:

- 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-MHP 1.0.3] clause 11.7.1.2, as included in [DVB-GEM 1.0.2] clause 11.7. Essentially, the above ensures platform stability when applications fail to follow the rules.

### 13.3.12 Application Discovery and Launching APIs

In addition to the APIs specified by [DVB-MHP 1.0.3] clause 11.7.2, as included in [DVB-GEM 1.0.2] 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.

### 13.3.13 Security

As a consequence of Section 6.1, OCAP extends [DVB-GEM 1.0.2], section 11.8 and includes the entire `java.security`, `java.security.spec`, and `java.security.cert` packages as defined by [JSR 217]. The `java.security.SecureClassLoader` SHALL derive from the PBP 1.1 `java.security.ClassLoader`.

### 13.3.14 Profile and Version Properties

In addition to the profile and version properties specified by [DVB-GEM 1.0.2] clause 11.9.3, OCAP reserves property names beginning with the string 'ocap'. OCAP terminals SHALL support the Profile and Version properties as per Table 13–4:

<table>
<thead>
<tr>
<th>Property</th>
<th>Semantics</th>
<th>Possible Values</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>ocap.profile</td>
<td>Name of the profile of the Host device. There are no active profiles at this time so the value in the field should be ignored.</td>
<td>&quot;OCAP 1.1&quot;</td>
<td>&quot;OCAP 1.1&quot;</td>
</tr>
<tr>
<td>ocap.version</td>
<td>Latest version of the OCAP profile supported by the Host device.</td>
<td>Concatenation of major, minor, and micro versions separated by &quot;.&quot;</td>
<td>&quot;1.2.0&quot;</td>
</tr>
<tr>
<td>ocap.version.major</td>
<td>Major version number of the latest OCAP profile supported by the Host device.</td>
<td>Non-negative integer value</td>
<td>&quot;1&quot;</td>
</tr>
<tr>
<td>ocap.version.minor</td>
<td>Minor version number of the latest OCAP profile supported by the Host device.</td>
<td>Non-negative integer value</td>
<td>&quot;2&quot;</td>
</tr>
<tr>
<td>ocap.version.micro</td>
<td>Micro version number of the latest OCAP profile supported by the Host device.</td>
<td>Non-negative integer value</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>ocap.version.update</td>
<td>List of ECNs against the version of the OCAP profile implemented by this Host that are supported by this Host.</td>
<td>Null, or an empty string, or a string that starts with &quot;:&quot; followed by a &quot;:&quot;-separated list of decimal ECN identifiers and ending with a &quot;:&quot;</td>
<td>&quot;:1118:1119:&quot;</td>
</tr>
</tbody>
</table>

#### 13.3.14.1 Information on Options (informative)

[DVB-MHP 1.0.3] clause 11.9.3.1, as included in [DVB-GEM 1.0.2] clause 11.9.3, defines properties for optional features. As with MHP and GEM, OCAP does not require any of the optional features.

#### 13.3.14.2 Optional API Support

In addition to optional properties specified by [DVB-MHP 1.0.3], 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.

Where an "ocap.api.option." property is defined, an additional property SHALL be defined with ".update" appended to its name (e.g., "dvr.update"). The property SHALL contain the release identifier of the corresponding specification (e.g., "I03") followed by a list of applied ECNs using the form specified in Table 13–4 for the "ocap.version.update" property (e.g., "I03:1027:1048:").

13.3.14.3 System Properties

This section extends [DVB-GEM 1.0.2] Section 11.9.3.

OCAP defines the following system properties as per Table 13–5:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Value</th>
<th>Application Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ocap.cablecard.manufacturer</td>
<td>CableCARD manufacturer identifier (see Table 9.5-4, [CCIF]). Same value returned from org.ocap.hardware.pod.POD.getManufacturerID method. Use the org.ocap.hardware.pod.POD.isReady method to determine if this value is valid.</td>
<td>Short Range: 0&lt;=value&lt;=0xFFFF</td>
<td>unsigned and signed</td>
</tr>
<tr>
<td>ocap.cablecard.version</td>
<td>Manufacturer version of the CableCARD (see Table 9.5-4, [CCIF]). Same value returned from org.ocap.hardware.pod.POD.getVersionNumber method. Use the org.ocap.hardware.pod.POD.isReady method to determine if this value is valid.</td>
<td>Short Range: 0&lt;=value&lt;=0xFFFF</td>
<td>uong and signed</td>
</tr>
<tr>
<td>ocap.cablecard.identifier</td>
<td>The 30 least significant bits of the Card_ID field from the CableCARD certificate (see section 5.5, [OC-SEC]).</td>
<td>Long Range: 0&lt;=value&lt;=999999999</td>
<td>MonitorAppPermission(&quot;properties&quot;)</td>
</tr>
<tr>
<td>ocap.cablecard.vct-id</td>
<td>VCT-ID derived from CableCARD. Same value returned from vct-id generic feature code 0x0D.</td>
<td>Int Range: -1&lt;=value&lt;=0xFFFF -1 == no value set</td>
<td>unsigned and signed</td>
</tr>
<tr>
<td>ocap.hardware.vendor_id</td>
<td>Vendor id- the Organizationally Unique Identifier (OUI) for the Host device manufacturer. (See IEEE OUI listing2.)</td>
<td>Int Range: 0&lt;value&lt;2^24</td>
<td>MonitorAppPermission(&quot;properties&quot;)</td>
</tr>
<tr>
<td>ocap.hardware.version_id</td>
<td>Unique Hardware identifier assigned to each version of a Host device from a particular vendor for purposes of Common download. (See [CDL] section 6.2.2 hardware_version_id.)</td>
<td>Long Range: 0&lt;value&lt;2^32</td>
<td>MonitorAppPermission(&quot;properties&quot;)</td>
</tr>
<tr>
<td>ocap.hardware.version</td>
<td>Manufacturer version number of the Host device3.</td>
<td>String</td>
<td>MonitorAppPermission(&quot;properties&quot;)</td>
</tr>
<tr>
<td>ocap.hardware.model_id</td>
<td>Manufacturer model number of the Host device.</td>
<td>String</td>
<td>MonitorAppPermission(&quot;properties&quot;)</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td>Value</td>
<td>Application Access</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------</td>
<td>---------------------------------------------------------</td>
</tr>
<tr>
<td>ocap.hardware.createdate</td>
<td>Manufacture date of the Host device, formatted as “MM-DD-YYYY.”</td>
<td>String</td>
<td>MonitorAppPermission(&quot;properties&quot;)</td>
</tr>
<tr>
<td>ocap.hardware.serialnum</td>
<td>Manufacturer serial number of the Host device.</td>
<td>String</td>
<td>MonitorAppPermission(&quot;properties&quot;)</td>
</tr>
<tr>
<td>ocap.hardware.host_id</td>
<td>Host Identifier; see org.ocap.hardware.Host.getId method.</td>
<td>String</td>
<td>unsigned and signed</td>
</tr>
<tr>
<td>ocap.software.vendor_id</td>
<td>Vendor id - the Organizationaly Unique Identifier (OUI) for the OCAP implementation manufacturer. (See IEEE OUI listing2).</td>
<td>Int Range: 0&lt;value&lt;2^24</td>
<td>MonitorAppPermission(&quot;properties&quot;)</td>
</tr>
<tr>
<td>ocap.software.version</td>
<td>Manufacturer version of the OCAP implementation3.</td>
<td>String</td>
<td>MonitorAppPermission(&quot;properties&quot;)</td>
</tr>
<tr>
<td>ocap.software.model_id</td>
<td>Manufacturer model number of the OCAP implementation.</td>
<td>String</td>
<td>MonitorAppPermission(&quot;properties&quot;)</td>
</tr>
<tr>
<td>ocap.memory.video1</td>
<td>Separate video memory in KB.</td>
<td>Long Range: 0&lt;value</td>
<td>MonitorAppPermission(&quot;properties&quot;)</td>
</tr>
<tr>
<td>ocap.memory.total1</td>
<td>Total DRAM in KB - represents footprint of device and not memory available to applications.</td>
<td>Long Range: 0&lt;value</td>
<td>MonitorAppPermission(&quot;properties&quot;)</td>
</tr>
<tr>
<td>ocap.system.highdef</td>
<td>Support for high definition.</td>
<td>boolean</td>
<td>unsigned and signed</td>
</tr>
<tr>
<td>ocap.api.option.*</td>
<td>Support for read access to OCAP API extensions</td>
<td>varies</td>
<td>unsigned and signed</td>
</tr>
</tbody>
</table>

1 The properties ocap.memory.video and ocap.memory.total are intended to be used in conjunction with ocap.hardware.vendor_id and ocap.hardware.version_id to identify receivers that are otherwise identical except for differing memory sizes. Since the different receiver implementation will use memory in different ways, they are meaningless to use under other circumstances.

2 The integer value of an OUI can be converted to the triple octet value defined by IEEE.

3 A manufacturer specific version number SHOULD contain a major and minor version number and MAY contain a micro value. The implementation SHALL use a period "." separator in between the major, minor, micro values, e.g., "2.3a.b1". In this case the order of the alpha-numerical terms is major, minor, micro version numbers. When a third separator is used, any sub-string that follows it is implementation specific. When an implementation does not use one of the version terms, but does use an implementation specific term, it MAY fill in the version term with a 0, e.g., "2.3a.0.xyz".

4 The ocap.hardware.version_id is specific to the definition in [CDL] and the ocap.hardware.version is a vendor specific version number.

13.3.15 Java Permissions

13.3.15.1 org.dvb.net.rc.RCPermission (informative)

[DVB-MHP 1.0.3] clause 11.10.2.5, as included in [DVB-GEM 1.0.2] 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.
13.3.15.2 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.)

13.3.15.3 java.util.PropertyPermission

This section extends [DVB-GEM 1.0.2] Section 11.10, which refers to [DVB-MHP 1.0.3] Section 11.10.2.1.

For all applications, a read permission shall be granted for all properties defined in Section 13.3.14.3 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.

13.3.15.4 java.net.SocketPermission

OCAP extends [DVB-MHP 1.0.3], Section 11.10.2.9 as included by [DVB-GEM 1.0.2], 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.

13.3.16 MHP 1.1 and OCAP

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] section. OCAP complies with section 11.7.9 of [DVB-MHP 1.1.3] for inclusion of the DVB service provider interface (SPI) packages org.dvb.spi, org.dvb.spi.selection, and org.dvb.spi.util with the following extensions:

1. The org.dvb.selection.SelectionProviderContext.serviceDescriptionAvailable method description SHALL be considered to include the following statement, "Services passed to this method that were not also passed to the getServiceList or serviceListChanged methods SHALL be ignored successfully."

2. The org.dvb.spi.XletBoundProvider.getBoundPBPXletContext method SHALL NOT be included.

13.3.17 JSR 280 XML Packages

These APIs are specified in [JSR 280]. Packages org.w3c.dom.events and org.w3c.dom.views SHALL be supported.
13.3.18 Logger API

A subset of the log4j distribution is required. The log4j distribution can be obtained at http://logging.apache.org/log4j. The following Java classes and interfaces from the log4j distribution SHALL be supported:

- `org.apache.log4j.Appender`
- `org.apache.log4j.AsyncAppender`
- `org.apache.log4j.AppenderSkeleton`
- `org.apache.log4j.Category`
- `org.apache.log4j.Hierarchy`
- `org.apache.log4j.Layout`
- `org.apache.log4j.Level`
- `org.apache.log4j.LogManager`
- `org.apache.log4j.Logger`
- `org.apache.log4j.PatternLayout`
- `org.apache.log4j.Priority`
- `org.apache.log4j.PropertyConfigurator`
- `org.apache.log4j.config.PropertySetter`
- `org.apache.log4j.or.ObjectRenderer`
- `org.apache.log4j.or.RenderMap`
- `org.apache.log4j.spi.AppenderAttachable`
- `org.apache.log4j.spi.Configurator`
- `org.apache.log4j.spi.ErrorHandler`
- `org.apache.log4j.spi.Filter`
- `org.apache.log4j.spi.HierarchyEventListener`
- `org.apache.log4j.spi.LocationInfo`
- `org.apache.log4j.spi.LoggerFactory`
- `org.apache.log4j.spi.LoggerRepository`
- `org.apache.log4j.spi.OptionHandler`
- `org.apache.log4j.spi.RolloverStrategy`
- `org.apache.log4j.spi.LoggerRepositorySelector`
- `org.apache.log4j.spi.ThrowableInformation`
- `org.apache.log4j.spi.ThrowableRenderer`
- `org.apache.log4j.spi.ThrowableRendererSupport`
- `org.apache.log4j.xml.DOMConfigurator`

Note: Implementations MAY include other files from the log4j distribution in an implementation-specific fashion.

OCAP extends the log4j distribution and modifies the following files:

- `org.apache.log4j.Category`
- `org.apache.log4j.Hierarchy`
- `org.apache.log4j.Logger`
- `org.apache.log4j.LogManager`
- `org.apache.log4j.PropertyConfigurator`
- `org.apache.log4j.config.PropertySetter`
- `org.apache.log4j.or.ObjectRenderer`
- `org.apache.log4j.xml.DOMConfigurator`

The javadoc for the log4j files modified by this specification can be seen in Annex AA.

In order to output log events to a MIB table, the implementation SHALL support the `org.ocap.logging.MIBAppender` class defined in Annex AA.

13.3.18.1 Configuration Considerations

13.3.18.1.1 Event properties

If a logging method was called from an application context, the xlet name, app id, and org id are added as event properties. If a Logger was retrieved using the form taking a group name, the group name is also added as an event property.
Event properties can be included in a PatternLayout definition using the following conversion patterns:

<table>
<thead>
<tr>
<th>Property name</th>
<th>Conversion pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>group</td>
<td>%X{group}</td>
</tr>
<tr>
<td>xlet-name</td>
<td>%X{xlet-name}</td>
</tr>
<tr>
<td>app-id</td>
<td>%X{app-id}</td>
</tr>
<tr>
<td>org-id</td>
<td>%X{org-id}</td>
</tr>
</tbody>
</table>

13.3.18.1.2 Appending events asynchronously

By default, a logging method will construct the logging event and write the event to appenders synchronously on the calling thread. AsyncAppender can be used to move the writing of the event to appenders onto a separate thread.

To use an AsyncAppender, define an AsyncAppender in an xml-based log4j configuration file (an AsyncAppender cannot be defined using a properties file-based log4j configuration file), and nest the target appender definition inside the AsyncAppender definition.

The AsyncAppender queue size is configurable, as is AsyncAppender’s behavior when the queue is full, including discard events with a summary message, and synchronously append events. Javadoc for AsyncAppender is available here:

http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/AsyncAppender.html

13.3.18.1.3 Filters

Filters are log4j’s mechanism supporting fine-grained control over whether or not an event is processed by an appender. Filters are attached to appenders, and using And Filter can be chained and nested. An appender will process an event only if the Filter chain allows it.

Filters can access all fields of a logging event to determine if the event should be processed, including the group, xlet name, and app id and org id if available.

NOTE: The Filter mechanism evaluates LoggingEvent objects. Creation of a LoggingEvent object includes the overhead required to build the event’s message including any string concatenation. This overhead may be a concern in performance-sensitive configurations.

13.3.18.1.4 Optional features

Some of the classes mentioned below, i.e., EnhancedPatternLayout, and ExpressionFilter, may not be available, as they are not required. The following sections provide information on these potentially useful optional features.

13.3.18.1.4.1 Enhanced Pattern layouts

If available, use of EnhancedPatternLayout is encouraged, which provides more flexible pattern support and format modifiers. Javadoc for EnhancedPatternLayout is available here:


13.3.18.1.4.2 ExpressionFilter syntax

ExpressionFilter is a very flexible Filter supporting logical operators, case-insensitive partial-text match support, grouping, negation and optional regular expression support. Javadoc for ExpressionFilter is available here:
An expression is a textual representation of a rule, and provides the ability to perform queries against a collection of logging events. As each logging event is examined by the rule, the operation associated with the rule is performed providing the event passes the rule's evaluation. Log4J's rule expression syntax provides access to all of the logging event's fields, a number of operators, and the ability to control precedence.

- Precedence can be controlled by applying parentheses to groups of expressions, usually used in combination with logical operators
- Single tick marks can be used to delimit multi-word operands
- Example: msg =~ 'my msg'

**Define an expression:**

1. Select the field identifier (a keyword representing a logging event field) required to perform the evaluation. **Field identifiers are not case sensitive.** See the list of supported field identifiers below.
2. Example: use the MSG field identifier in an expression to evaluate the 'message' field of the logging event.
3. Select the operator needed to perform the evaluation. See the list of supported operators below.
4. Example: to perform a case-insensitive partial-text evaluation, use the =~ operator
5. Define the value to be matched in the expression (required by all operators except logical operators).
6. Example: to display only events which are greater than or equal to level INFO, use the expression LEVEL >= INFO

The field identifier/operator/operand combination itself is a valid expression, but can be used in combination with other expressions and logical operators (&&, ||) in order to define complex expressions

Modify the precedence of expressions in a complex expression by surrounding the expression(s) in parentheses

**Supported field identifiers:**

- LOGGER
- LEVEL
- CLASS
- FILE
- LINE
- METHOD
- MSG
- NDC
- EXCEPTION
- TIMESTAMP
- THREAD
- PROP.[key name]

**NOTE:** To build expressions using PROP (properties) field identifier, the key being evaluated must be added after the field identifier and a period.
Example: to only append events associated with app id 12345, use the expression: PROP.app-id = 12345

**Supported operators:**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>equals</td>
</tr>
<tr>
<td>~=</td>
<td>case-insensitive partial-text match</td>
</tr>
<tr>
<td>!=</td>
<td>not equals</td>
</tr>
<tr>
<td>LIKE</td>
<td>supports regular expressions</td>
</tr>
<tr>
<td>EXISTS</td>
<td>not null</td>
</tr>
<tr>
<td>&lt;</td>
<td>inequality operators</td>
</tr>
<tr>
<td>&lt;=</td>
<td></td>
</tr>
<tr>
<td>&gt;</td>
<td></td>
</tr>
<tr>
<td>&gt;=</td>
<td></td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>logical and</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>!</td>
<td>logical not</td>
</tr>
</tbody>
</table>

**Expression shortcut:**

- Since it is very common need to perform a case-insensitive partial-text match against the Message field, any single word or any single phrase (the phrase must be delimited by single-quote marks) will act as a rule defined using the `MSG ~=` syntax.

- Example: to search the Message field for the phrase 'logged in', instead of defining the expression `MSG ~= 'logged in'`, use `'logged in'` by itself (including the single quotes). Instead of using `MSG ~= evaluation` to search for the word 'evaluation', use `evaluation` by itself (no need for single quotes since it's a single word).

- **NOTE:** This shortcut syntax is designed to be used to search for **single words or phrases only**, and will not work when used in combination with other operators or expressions. When building complex expressions, use the `MSG ~=` syntax.

**Operator limitations:**

- Inequality operators are supported for Levels (`LEVEL > info`) and anything that can be converted to a numeric value (including timestamps).

- In order to build expressions using the Timestamp field, provide the Timestamp value in this format: 'yyy/MM/dd HH:mm:ss'. Note the single ticks, which are required because of the space between the days and hours. The Timestamp field contains millisecond information, but the milliseconds are rounded off during event evaluation.

**Example expressions:**

<table>
<thead>
<tr>
<th>To display all events with a level of info or greater</th>
<th>LEVEL &gt;= INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>To display all events which contain the words 'logged in' in the message</td>
<td>MSG ~= 'logged in'</td>
</tr>
<tr>
<td>To display all events which contain an exception</td>
<td>EXCEPTION EXISTS</td>
</tr>
</tbody>
</table>
To display all events associated with a PROP key of USERID with a value of 'tester'

\[ \text{PROP.USERID == tester} \]

'and' the first and second examples together, 'or'd with the third

\[ ( \text{LEVEL} \geq \text{INFO} \&\& \text{MSG} \sim \text{'logged in'} \) \| \text{EXCEPTION EXISTS} \]

To display all events between 23:55 and 23:56 on the 19th of June

\[ \text{TIMESTAMP} \geq '2004/06/19 23:55:00' \&\& \text{TIMESTAMP} \leq '2004/06/19 23:56:00' \]

To display events occurring at 23:55:12 on the 19th of June (all events generated during that second, regardless of millisecond value)

\[ \text{TIMESTAMP} \sim '2004/06/19 23:55:12' \]

### 13.3.18.2 Porting log4j to PBP 1.1

The log4j implementation was created for J2SE and as such requires some changes to compile and run in a PBP 1.1 environment. For example, org.apache.log4j.config.PropertySetter was changed to use reflection instead of javabeans introspection. PBP 1.1 porting changes do not require changes to class, interface, or method definitions, or changes to method signatures. Consequently, definition of the porting changes is out of scope of this specification.

### 13.3.19 System Logging

The implementation performs logging when directed by the control MIB ocStbHostSystemLoggingControl; see [MIB-HOST]. A network management system can write the ocStbHostSystemLoggingControl and enable system logging based on group and level. The set of groups includes:

- "All"
- "Performance.ServiceSelection"
- "Performance.Tuning"

The set of levels in increasing severity order includes:

1. ALL
2. TRACE
3. DEBUG
4. INFO
5. WARN
6. ERROR
7. FATAL
8. OFF

Setting a level sets all levels from the level specified to the highest severity level except for "OFF". The group "All" indicates all groups. The level "ALL" indicates all levels.

#### 13.3.19.1 System Log Events

This section defines System Log Events for mandatory system logging. The implementation SHALL log System Log Events when the group and level of an event definition matches a MIB-enabled group and level. The events will be logged as a ocStbHostSystemLoggingEntryMessage in the ocStbHostSystemLoggingTable as defined in [MIB-HOST].

Setting the ocStbHostSystemLoggingEntryMessage string is specified below SHALL control the output format. The System Log Events defined below SHALL be created using a pattern layout of:
"[%r] (%g-%l) %m%n"

In this case the pattern layout conversion specifiers are defined as follows:

- %r Used to output the number of milliseconds elapsed from January 1, 1970 until the trigger of a System Log Message.
- %g Used to output the group of the logging event.
- %l Used to output the level of the logging event.
- %m Used to output the application supplied message associated with the logging event.
- %n Outputs a newline.

Following is an example log entry when event is defined as Group:"Performance.ServiceSelection", Level:"INFO" and Message: "Presentation Started":

[123456789] (Performance.ServiceSelection-INFO) Service Selection Started: ServiceContext-0x12345678, Locatorocap://0x45a

A field within angle brackets < > is a value that is converted to a sub-string in the message. Some messages include a <locator>, <service context id>, or <tuner id> field. The <locator> field SHALL be compliant with the OCAP URL definition as defined in Section 16. The <tuner id> and <service context id> field SHALL uniquely identify the ServiceContext or tuner and SHALL comply with the "unreserved" definition from Section 16. The <frequency>, <program_number>, and <modulation_format> fields SHALL conform to Table 16–4.

13.3.19.1.1 Service Selection Started Event

Trigger: Service selection begins, e.g., a ServiceContext.select method is called.

Group: "Performance.ServiceSelection"

Level: "INFO"

Message: "Service Selection Started:ServiceContext <service context id>, Locator <locator>"

13.3.19.1.2 Tuner Acquisition Started

Trigger: When Tuner resource acquisition is started for a ServiceContext.

Group: "Performance.ServiceSelection"

Level: "INFO"

Message: "Tuner Acquisition Start: ServiceContext <service context id>"

13.3.19.1.3 Tuner Acquisition Complete

Trigger: When a Tuner resource is acquired for a ServiceContext.

Group: "Performance.ServiceSelection"

Level: "INFO"
Message: "Tuner Acquisition Complete: ServiceContext <service context id>, Tuner <tuner id>"

13.3.19.1.4 Tuning Initiated

Trigger: When the ServiceContext initiates tuning.

Group: "Performance.ServiceSelection"

Level: "INFO"

Message: "Tuning Initiated: ServiceContext <service context id>, Tuner <tuner id>"

13.3.19.1.5 Media Access Handler Invoked

Trigger: When the MediaAccessHandler is invoked.

Group: "Performance.ServiceSelection"

Level: "INFO"

Message: "Media Access Handler Invoked: ServiceContext <service context id>"

13.3.19.1.6 Media Access Handler Returned

Trigger: When a Tuner resource is acquired for a ServiceContext.

Group: "Performance.ServiceSelection"

Level: "INFO"

Message: "Media Access Handler Returned: ServiceContext <service context id>, <fully|partially|not> authorized"

13.3.19.1.7 Conditional Access Requested Event

Trigger: When Host initiates a CA request.

Group: "Performance.ServiceSelection"

Level: "INFO"

Message: "CA Requested: ServiceContext <service context id>"

13.3.19.1.8 Conditional Access Granted Event

Trigger: When Host receives a ca_pmt APDU from CableCARD and the ca_pmt_cmd_id is set to ok_descrambling.

Group: "Performance.ServiceSelection"

Level: "INFO"

Message: "CA Granted: ServiceContext <service context id>, CA session <ca session id>"
13.3.19.1.9 Decode Initiated

Trigger: When decoding of the service has begun.

Group: "Performance.ServiceSelection"

Level: "INFO"

Message: "Decode: ServiceContext <service context id>"

13.3.19.1.10 Video Presented Event

Trigger: When a new service is selected, video presentation begins, and a javax.tv.service.selection.NormalContentEvent is generated.

Group: "Performance.ServiceSelection"

Level: "INFO"

Message: "Presentation Started: ServiceContext <service context id>"

13.3.19.1.11 Tuning Started Event

Trigger: The tuning operation has started.

Group: "Performance.Tuning"

Level: "INFO"

Message: "Tune Started: Tuner <tuner id>,f <frequency>,p <program_number>,m <modulation_format>"

13.3.19.1.12 QAM Lock Event

Trigger: QAM lock occurs.

Group: "Performance.Tuning"

Level: "INFO"

Message: "QAM Lock: Tuner <tuner id>"

13.3.19.1.13 PAT Acquired Event

Trigger: Program Allocation Table (PAT) completely acquired from the network.

Group: "Performance.Tuning"

Level: "INFO"

Message: "PAT Acquired: Tuner <tuner id>"
13.3.19.1.14 PMT Acquired Event

Trigger: The Program Map Table (PMT) for the currently-selected program is completely acquired from the network. In the case of a transport stream tune, this will be triggered by acquisition of the first PMT.

Group: "Performance.Tuning"

Level: "INFO"

Message: "PMT Acquired: Tuner <tuner id>, program <program_number>"

13.4 Full Java API List (informative)

This subsection does not correspond to any [DVB-GEM 1.0.2] Section.

Section 11 in [DVB-GEM 1.0.2] does require, with some modification, the Java packages that are detailed in [DVB-MHP 1.0.3]. 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-MHP 1.0.3] as included in and modified by [DVB-GEM 1.0.2] 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

The following list of packages required by OCAP are included as defined by [JSR 217].

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
java.net
java.rmi
java.rmi.registry
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.microedition.xlet
javax.microedition.xlet.ixc
javax.security.auth.x500
13.4.2 Java TV 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-MHP 1.0.3].

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 1.0.3 Packages

The following list of packages required by OCAP excludes org.dvb.si from the [DVB-MHP 1.0.3] list of packages:

org.dvb.application
org.dvb.damcc
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 F)
org.ocap.system (see Annex Q)
org.ocap.ui (see Annex E)
org.ocap.ui.event (see Annex E)
org.ocap.ai (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.diagnostics (see Annex Z)
org.ocap.environment (see Annex Y)

13.4.9 MHP 1.1 Packages

org.dvb.spi
org.dvb.spi.selection
org.dvb.spi.util

NOTE: The packages above are defined in [DVB-MHP 1.1.3] Annex AN.

org.dvb.service.selection

NOTE: The package above is defined in [DVB-MHP 1.1.3] Annex AK.

13.4.10 Logger Packages

org.apache.log4j
org.apache.log4j.config
org.apache.log4j.spi
org.apache.log4j.xml
org.ocap.logging

NOTE: Not all of the log4j classes and interfaces are required; see Section 13.3.18 for a list of the required log4j
classes and interfaces.

13.5 GEM and OCAP

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

13.5.1 VK_TELETEXT Key

OCAP 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-MHP 1.0.3] clause G.5 does not apply to
OCAP. See also [DVB-GEM 1.0.2] clause G as modified by Section 6.1.1.1, and [DVB-MHP 1.0.3] clause
11.4.1.4, which is included in [DVB-GEM 1.0.2] 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 1.0.2].
13.5.3 Locale Support (informative)

OCAP requires support for locales in addition to those required by [DVB-GEM 1.0.2]. Locales supported in OCAP are specified in Section 17.2.1.2.

**NOTE:** [DVB-MHP 1.0.3] clause 11.3.1.3, which is included in [DVB-GEM 1.0.2] 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, 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 uses org.ocap.event.EventManager which extends org.dvb.event.EventManager.

The OCAP user input event API as specified in Annex K works through the org.dvb.event package. The OCAP package is org.ocap.event.

**NOTE:** org.dvb.event is as specified in [DVB-MHP 1.0.3] clause 11.3.2.2, which is included in [DVB-GEM 1.0.2] clause 11.3.
14 SECURITY

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 12 Security and are extensions of [DVB-MHP 1.0.3] Section: 12 Security.

This OCAP chapter indicates in which ways security on an OCAP platform conforms to security on a [DVB-MHP 1.0.3] 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 1.0.2]. OCAP applications SHALL only include those security files defined in this specification. 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.

NOTE: In addition to [DVB-MHP 1.0.3] section 12 and this section, application authentication rules are also found in Section 12.2.7 of this specification. Primary considerations for application authentication are found in [DVB-MHP 1.0.3] 12.4.1.4, 12.6.1, 12.7, and Sections 14.2.1.16 and 14.2.2.2 of this specification.

14.1 DVB-GEM and DVB-MHP Specification Correspondence

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section. Section 14 (this section) of OCAP corresponds to Section 12 of [DVB-MHP 1.0.3] and Section 12 of [DVB-GEM 1.0.2] as follows:

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>[DVB-GEM 1.0.2] Compliance</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>[DVB-MHP 1.0.3] Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>14 Security</td>
<td>12 Security</td>
<td>Extension</td>
<td>12 Security</td>
<td>Extension</td>
</tr>
<tr>
<td>14.1 DVB-GEM and DVB-MHP Specification</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>14.2 OCAP Specific Requirements</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>14.2.1 Deviations from the DVB-MHP</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>Specification</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.1 Introduction</td>
<td>A subsection is extended</td>
<td>12.1 Introduction</td>
<td>A subsection is extended</td>
</tr>
<tr>
<td>14.2.1.1 Overview of the Security Framework for Applications</td>
<td>12.1.1 Overview of the security framework for applications</td>
<td>Extension</td>
<td>12.1.1 Overview of the security framework for applications</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.1.2 Overview of return channel security</td>
<td>Compliance</td>
<td>12.1.2 Overview of return channel security</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.1.3 Extensions to MHP application signing framework</td>
<td>Compliance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.2 Authentication of applications</td>
<td>Compliance</td>
<td>12.2 Authentication of applications</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.3 Message transport</td>
<td>Compliance</td>
<td>12.3 Message transport</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.4 Detail of application authentication messages</td>
<td>A subsection is extended</td>
<td>12.4 Detail of application authentication messages</td>
<td>A subsection is extended</td>
</tr>
<tr>
<td>OCAP</td>
<td>[DVB-GEM 1.0.2] Section</td>
<td>GEM Compliance</td>
<td>[DVB-MHP 1.0.3] Section</td>
<td>MHP Compliance</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.1 Hash File</td>
<td>A subsection is extended</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.1.1 Description</td>
<td>Compliance</td>
</tr>
<tr>
<td>14.2.1.2 HashFile Location and Naming Conventions</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.1.2 HashFile location and naming conventions</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.1.3 Digest value computation rules</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.1.4 Warning concerning grouping of objects under a single digest (Informative)</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.1.5 Special authentication rules</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.2 Signature File</td>
<td>A subsection is extended</td>
</tr>
<tr>
<td>14.2.1.3 SignatureFile Location and Naming Conventions</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.2.1 Description</td>
<td>Extension</td>
</tr>
<tr>
<td>14.2.1.3 SignatureFile Location and Naming Conventions</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.2.2 SignatureFile location and naming conventions</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.2.3 Supported algorithms</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.2.4 Signature computation rules</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.2.5 Authentication rules</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.3 Certificate File</td>
<td>A subsection is extended</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.3.1 Description</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.3.2 ASN.1 encoding</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.3.3 Supported algorithms</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.3.4 Name matching</td>
<td>Compliance</td>
</tr>
<tr>
<td>14.2.1.4 CertificateFile Location and Naming Conventions</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.3.5 CertificateFile location and naming conventions</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.3.6 Authentication rules</td>
<td>Compliance</td>
</tr>
<tr>
<td>14.2.1.16 Integration of the File Authentication Process</td>
<td>12.4 Detail of application authentication messages</td>
<td>Compliance</td>
<td>12.4.4 Integration</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.5 Profile of X.509 certificates for authentication of applications</td>
<td>Compliance</td>
<td>12.5 Profile of X.509 certificates for authentication of applications</td>
<td>Extension</td>
</tr>
<tr>
<td>OCAP</td>
<td>[DVB-GEM 1.0.2] Section</td>
<td>GEM Compliance</td>
<td>[DVB-MHP 1.0.3] Section</td>
<td>MHP Compliance</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
<td>----------------</td>
<td>---------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.5 Profile of X.509 certificates for authentication of applications</td>
<td>Compliance</td>
<td>12.5.1 signatureAlgorithm</td>
<td>Compliance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12.5.2 signatureValue</td>
<td>Compliance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12.5.3 version</td>
<td>Compliance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12.5.4 issuer</td>
<td>Compliance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12.5.5 validity</td>
<td>Compliance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12.5.6 subject</td>
<td>Compliance</td>
</tr>
<tr>
<td>14.2.1.6 subjectPublicKeyInfo</td>
<td>12.5 Profile of X.509 certificates for authentication of applications</td>
<td>Compliance</td>
<td>12.5.7 SubjectPublic Key Info</td>
<td>Extension</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.5 Profile of X.509 certificates for authentication of applications</td>
<td>Compliance</td>
<td>12.5.8 Unique Identifiers</td>
<td>Compliance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.5 Profile of X.509 certificates for authentication of applications</td>
<td>Compliance</td>
<td>12.5.9 Extensions</td>
<td>Extension</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.6 Security policy for applications</td>
<td>A subsection is extended</td>
<td>12.6 Security policy for applications</td>
<td>A subsection is extended</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.6.1 General principles</td>
<td>Extension</td>
<td>12.6.1 General principles</td>
<td>Extension</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.6.2 Permission request file</td>
<td>Compliance</td>
<td>12.6.2 Permission request file</td>
<td>Compliance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.6.2.1 File encoding</td>
<td>Compliance</td>
<td>12.6.2.1 File encoding</td>
<td>Compliance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.6.2.2 File integrity</td>
<td>Compliance</td>
<td>12.6.2.2 File integrity</td>
<td>Compliance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.6.2.3 Example</td>
<td>Extension</td>
<td>12.6.2.3 Example</td>
<td>Extension</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.6.2.4 Permission request file name and location</td>
<td>Extension</td>
<td>12.6.2.4 Permission request file name and location</td>
<td>Extension</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.6.2.5 Permission request file</td>
<td>Compliance</td>
<td>12.6.2.5 Permission request file</td>
<td>Compliance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.6.2.6 Credentials</td>
<td>Compliance</td>
<td>12.6.2.6 Credentials</td>
<td>Compliance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.6.2.7 File Access</td>
<td>Compliance</td>
<td>12.6.2.7 File Access</td>
<td>Compliance</td>
</tr>
<tr>
<td>OCAP</td>
<td>[DVB-GEM 1.0.2] Section</td>
<td>GEM Compliance</td>
<td>[DVB-MHP 1.0.3] Section</td>
<td>MHP Compliance</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------</td>
<td>----------------</td>
<td>-------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.6.2.8 CA API</td>
<td>Compliance</td>
<td>12.6.2.8 CA API</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.6.2.9 Application lifecycle control policy</td>
<td>Compliance</td>
<td>12.6.2.9 Application lifecycle control policy</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.6.2.10 Return channel access policy</td>
<td>Compliance</td>
<td>12.6.2.10 Return channel access policy</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.6.2.11 Tuning access policy</td>
<td>Compliance</td>
<td>12.6.2.11 Tuning access policy</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.6.2.12 Service selection policy for applications</td>
<td>Compliance</td>
<td>12.6.2.12 Service selection policy for applications</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.6.2.13 Media API access policy</td>
<td>Compliance</td>
<td>12.6.2.13 Media API access policy</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.6.2.14 Interapplication communication policy</td>
<td>Extension</td>
<td>12.6.2.14 Inter-application communication policy</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.6.2.15 User Setting and Preferences access policy</td>
<td>Compliance</td>
<td>12.6.2.15 User Setting and Preferences access policy</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.6.2.16 Network permissions</td>
<td>Compliance</td>
<td>12.6.2.16 Network permissions</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>No Corresponding Section</td>
<td></td>
<td>12.6.2.17 Dripfeed permissions</td>
<td>Compliance</td>
</tr>
<tr>
<td>14.2.1.11 Scenario Example</td>
<td>12.7 Example of creating an application that can be authenticated</td>
<td>Compliance</td>
<td>12.7 Example of creating an application that can be authenticated</td>
<td>Extension</td>
</tr>
<tr>
<td>12.8 GEM/MHP certification procedures</td>
<td>Compliance</td>
<td>12.8 MHP certification procedures</td>
<td>Compliance</td>
<td></td>
</tr>
<tr>
<td>14.2.1.13 Signature and Certificate Chain Validation</td>
<td>12.9 Certificate management</td>
<td>Extension</td>
<td>12.9 Certificate management</td>
<td>Extension</td>
</tr>
<tr>
<td>14.2.1.14 Security on the return channel</td>
<td>12.10 Security on the return channel</td>
<td>Extension</td>
<td>12.10 Security on the return channel</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.11 The internet profile of X.509 (informative)</td>
<td>A subsection is extended</td>
<td>12.11 The internet profile of X.509 (informative)</td>
<td>A subsection is extended</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.11 The internet profile of X.509 (informative)</td>
<td>A subsection is extended</td>
<td>12.11.2 Standard certificate extensions</td>
<td>A subsection is extended</td>
</tr>
<tr>
<td>14.2.1.15 SignatureFile Description</td>
<td>12.11 The internet profile of X.509 (informative)</td>
<td>Compliance</td>
<td>12.11.2.1 Authority key identifier</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.11 The internet profile of X.509 (informative)</td>
<td>Compliance</td>
<td>12.11.2.2 Subject key identifier</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.11 The internet profile of X.509 (informative)</td>
<td>Compliance</td>
<td>12.11.2.3 Key usage</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.11 The internet profile of X.509 (informative)</td>
<td>Compliance</td>
<td>12.11.2.4 Private key usage period</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.11 The internet profile of X.509 (informative)</td>
<td>Compliance</td>
<td>12.11.2.5 Certificate policies</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.11 The internet profile of X.509 (informative)</td>
<td>Compliance</td>
<td>12.11.2.6 Policy mappings</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>12.11 The internet profile of X.509 (informative)</td>
<td>Compliance</td>
<td>12.11.2.7 Subject Alternative Name</td>
<td>Compliance</td>
</tr>
</tbody>
</table>
### 14.2 OCAP Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

#### 14.2.1 Deviations from the DVB-MHP Specification

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

**14.2.1.1 Overview of the Security Framework for Applications**

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 12.1.1 Overview of the security framework for applications and are extensions of [DVB-MHP 1.0.3] Section: 12.1.1 Overview of the security framework for applications.

Section 12.1.1 of [DVB-GEM 1.0.2] applies to EE applications executing on OCAP. 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.

**14.2.1.2 HashFile Location and Naming Conventions**

This subsection is compliant with [DVB-GEM 1.0.2] Section: 12.4 Detail of application authentication messages and are extensions of [DVB-MHP 1.0.3] Section: 12.4.1.2 HashFile location and naming conventions.

This section extends Section 12.4 of [DVB-GEM 1.0.2], which references Section 12.4.1.2 of [DVB-MHP 1.0.3]. 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 1.0.2] Section: 12.4 Detail of application authentication messages and contains extensions of [DVB-MHP 1.0.3] Section: 12.4.2.1 Description and also contains requirements that are extensions to [DVB-MHP 1.0.3] Section: 12.4.2.2 SignatureFile location and naming conventions.

This section extends Section 12.4 of [DVB-GEM 1.0.2] and the corresponding Section 12.4.2.1 of [DVB-MHP 1.0.3] 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, OCAP complies with Section 12.4.2.2 of [DVB-MHP 1.0.3] corresponding to Section 12.4 of [DVB-GEM 1.0.2].

14.2.1.4 CertificateFile Location and Naming Conventions

This subsection is compliant with [DVB-GEM 1.0.2] Section: 12.4 Detail of application authentication messages and contains extensions of [DVB-MHP 1.0.3] 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 1.0.2] Section: 12.5 Profile of X.509 certificates for authentication of applications and contains extensions of [DVB-MHP 1.0.3] Section: 12.5.9 Extensions.

The referenced section of [DVB-GEM 1.0.2] 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 1.0.2] Section: 12.5 Profile of X.509 certificates for authentication of applications and contains extensions of [DVB-MHP 1.0.3] 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 1.0.2] Section: 12.6 Security policy for applications and contains extensions of [DVB-MHP 1.0.3] 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 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 1.0.2] Section.

An OCAP receiver SHALL support both the [DVB-MHP 1.0.3] 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 1.0.2].

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 1.0.2], may not have corresponding semantics in an OCAP environment. For clarification, an OCAP 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 1.0.2] Section: 12.6 Security policy for appl35 and contains extensions of [DVB-MHP 1.0.3] Section: 12.6.2.3 Example

Because OCAP has redefined several elements of the Permission Request File DTD, a different example from that used in Section 12.6.2.3 of [DVB-MHP 1.0.3], referred by 12.6 of [DVB-GEM 1.0.2] is necessary:

```xml
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE permissionrequestfile PUBLIC "//OCAP//DTD Permission Request File 1.0//EN" "http://www.opencable.com/ocap/dtd/ocappermissionrequestfile-1-0.dtd">
<!-- see MHP 1.0.3 section 12.6.2.1 and OCAP section 14.2.1.8 for details-->
<permissionrequestfile orgid="ffff" appid="6000">
  <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>
  <persistentfilecredential>
    <grantoridentifier id="ff"></grantoridentifier>
    <expirationdate date="27/04/2005"></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>
</permissionrequestfile>
<!-- Authentication information see MHP 1.0.3 section 12.6.2.6-->
<!-- various Permissions, see the OCAP spec for details-->
14.2.1.10 Permission Request File Name and Location

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 12.6 Security policy for applications and contains extensions of [DVB-MHP 1.0.3] 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-MHP 1.0.3].

14.2.1.11 Scenario Example

This subsection is compliant with [DVB-GEM 1.0.2] Section: 12.7 Example of creating an application that can be authenticated and contains extensions of [DVB-MHP 1.0.3] 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-MHP 1.0.3], referenced by 12.7 of [DVB-GEM 1.0.2] 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.

In addition, there are two changes in Section 12.7.2 of [DVB-MHP 1.0.3], referenced by 12.7 of [DVB-GEM 1.0.2], in order to correlate with this example and comply with OCAP:

- Change of Table 58, in which <ocap.Xlet1.perm> file SHALL be included;
- Change of Table 59, in which CertificateFile name SHALL be <ocap.certificates.1> instead of <dvb.certificates.1>.

14.2.1.12 OCAP Certification Procedures

This subsection contains OCAP-specific requirements that does not correspond to any [DVB-GEM 1.0.2] Section. This OCAP 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 1.0.2] Section: 12.9 Certificate management and contains extensions of [DVB-MHP 1.0.3] Section: 12.9 Certificate management.

This section extends Section 12.9 of [DVB-GEM 1.0.2] and corresponding Section 12.9.2 of [DVB-MHP 1.0.3].
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 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 Security on the return channel

This subsection contains requirements that are extensions to [DVB-MHP 1.0.3] Section12.10.2 Security on the return channel.

An OCAP implementation SHALL support the TLS cipher suites required by [DVB-MHP 1.0.3] section 12.10.2 except for the following suite:

- **TLS_RSA_EXPORT_WITH_DES40_CBC_SHA**

In addition to the TLS cipher suites required by [DVB-MHP 1.0.3], an OCAP implementation SHALL support the following suites:

- **TLS_RSA_WITH_AES_128_CBC_SHA**
- **TLS_RSA_WITH_AES_256_CBC_SHA**

The default order of the TLS cipher suites returned by the javax.net.ssl.SSLSocketFactory.getDefaultCipherSuites method SHALL be as follows:

- **TLS_RSA_WITH_AES_128_CBC_SHA** (in array index 0)
- TLS_RSA_WITH_AES_256_CBC_SHA (in array index 1)
- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_DES_CBC_SHA
- TLS_RSA_WITH_NULL_SHA
- TLS_RSA_WITH_NULL_MD5
- TLS_NULL_WITH_NULL_NULL

An OCAP implementation MAY support additional TLS cipher suites, but they SHALL NOT come before
TLS_RSA_WITH_AES_128_CBC_SHA or TLS_RSA_WITH_AES_256_CBC_SHA in the default cipher suite
order.

14.2.1.15 SignatureFile Description

This subsection is compliant with [DVB-GEM 1.0.2] Section: 12.11 The internet profile of X.509 (informative) and
contains extensions of [DVB-MHP 1.0.3] Section: 12.11.2.1 Authority key identifier.

OCAP extends section 12.1 of [DVB-GEM 1.0.2] and the corresponding Section 12.1.2.1 of [DVB-MHP 1.0.3] 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.

14.2.1.16 Integration of the File Authentication Process

OCAP modifies the [DVB-GEM 1.0.2] file authentication process in Section 12.4 and the corresponding Section
12.4.4 of [DVB-MHP 1.0.3], 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 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 1.0.2] which points to [DVB-MHP 1.0.3] 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. Dependent 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.

### 14.2.1.17 Inter-application communication policy

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 12.6.2.14 Inter-application communication policy, and contains extensions of [DVB-MHP 1.0.3] Section: 12.6.2.14 Inter-application communication policy.

The inter-application communication policy for an application SHALL be represented by the set of OcapIxcPermissions granted to that application (see 14.2.2.9).

NOTE: While the default policy is as described in [DVB-GEM 1.0.2] Section: 12.6.2.14 Inter-application communication policy, the policy may be expanded via the permission request file and constrained by both the SecurityPolicyHandler and the permission request file.

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

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

The following information is normative for OCAP. It is specific to OCAP and is not contained within the [DVB-GEM 1.0.2].
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-MHP 1.0.3] Permission Request File DTD since additional permissions have been added to meet OCAP requirements. However, in order to be compliant with the [DVB-GEM 1.0.2], the OCAP 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 1.3.1//EN"

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

http://www.opencable.com/ocap/dtd/ocappermissionrequestfile-1-3-1.dtd

The Name used in the document type declaration shall be "permissionrequestfile".

The OCAP PRF DTD is provided below:

```xml
<!-- ....................................................................... -->
<!-- OCAP Permission Request File DTD -->
<!-- ....................................................................... -->
<!-- file: ocappermissionrequestfile-1-3-1.dtd -->
<!-- ....................................................................... -->
<!-- This is the DTD for the OCAP 1.3.1 permission request file. The following formal public identifier SHALL be used to identify it: "-//OCAP//DTD Permission Request File 1.3.1//EN" The following URL for the SystemLiteral may be used to reference this file: http://www.opencable.com/ocap/dtd/ocappermissionrequestfile-1-3-1.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. -->
<!ATTLIST permissionrequestfile
    orgid CDATA #REQUIRED
    appid CDATA #REQUIRED
    xmlns:ocap CDATA #FIXED "ocap"
>
<!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 value (true | false) "true">
  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 name (registrar | service | servicemanager | security | reboot | systemevent | handler.appFilter | handler.resource | handler.closedCaptioning | filterUserEvents | handler.eas | setVideoPort | podApplication | signal.configured | properties | storage | registeredapi.manager | vbifiltering | codeDownload | mediaAccess | powerMode | environment.selection | logger.config) #IMPLIED>

<!ELEMENT ocap:registeredapi.user EMPTY>
<!ATTLIST ocap:registeredapi.user name CDATA #REQUIRED>

<!ELEMENT ocap:ixc EMPTY>
<!ATTLIST ocap:ixc scope CDATA #IMPLIED>
oid CDATA #IMPLIED
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. Monitor Application permissions SHALL only be granted to unbound applications. 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 value attribute is "true"; otherwise it is revoked. The type and action 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**

OCAP differs from the [DVB-GEM 1.0.2], 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 1.0.2].

In OCAP 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 1.0.2] are not used in OCAP to replace OpenCable root certificates.

14.2.2.8 **Logical Storage Volume Access**

The implementation's assessment of logical storage volume (LSV) 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.

a. If the application_id and organization_id of the accessing application match the corresponding fields in the LSV Extended File Access Permission, then the readApplicationAccessRight and writeApplicationAccessRight determine access to the LSV.

b. If the application_id of the accessing application does not match the corresponding field in the LSV, but the organization_id of the accessing application does match the corresponding field in the LSV Extended File Access Permission, then the readOrganizationAccessRight and writeOrganizationAccessRight determine access to the LSV.

c. If the organization_id of the accessing application does not match the corresponding field in the LSV, 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 readOtherOrganizationAccessRight and writeOtherOrganizationAccessRight determine access to the LSV.

d. If the organization does not match the organization_id or one of the other organization_id fields in the LSV,
then the `readWorldAccessRight` and `writeWorldAccessRight` determine access to the LSV.

### 14.2.2.9 OCAP IXC Permission

Access to inter-application communication SHALL be limited based upon the `OcapIxcPermission` (see Annex G) permissions that are granted to an application. Where the privilege necessary to perform a specific lookup operation is not granted, that lookup SHALL fail with a `java.rmi.NotBoundException`. Where the privilege necessary to perform a specific bind operation is not granted, that bind SHALL fail silently with no other side effects.

#### 14.2.2.9.1 Unsigned applications

By default, an unsigned application SHALL be granted the following `OcapIxcPermissions`:

- `OcapIxcPermission("/service-\*/unsigned/\*/\*/\", "lookup")`
- `OcapIxcPermission("/service-service\_context_id/unsigned/organization\_id/application\_id/\", "bind")`

Organization_id and application_id refer to the organization_id and application_id of the calling xlet (see [DVB-MHP 1.0.3] Section 10.5.1), and shall be formatted as specified for `org.dvb.io.ixc.IxcRegistry.lookup()`. Service_context_id is a unique platform-generated opaque name that identifies the service context for the purpose of same-service scoping of inter-xlet communication (as specified for signed applications in [DVB-MHP 1.0.3] Section: 12.6.2.14 Inter-Application Communication policy). The same name shall be used by all applications running in that service context.

This is sufficient to grant an unsigned application the rights for Inter-Application Communication as specified in [DVB-MHP 1.0.3] Section: 12.6.2.14 Inter-Application Communication policy.

The installed `SecurityPolicyHandler`, if there is one, MAY deny these default permissions outright or grant only a subset of these permissions (see 21.2.1.9).

#### 14.2.2.9.2 Signed applications

By default, a signed application SHALL be granted the following `OcapIxcPermissions`:

- `OcapIxcPermission("/service-service\_context_id/signed/\*/\*/\", "lookup")`
- `OcapIxcPermission("/service-service\_context_id/signed/organization\_id/application\_id/\", "bind")`

Organization_id, application_id, and service_context_id are as defined for unsigned applications in 14.2.2.9.1.

This is sufficient to grant a signed application the rights for Inter-Application Communication as specified in [DVB-MHP 1.0.3] Section: 12.6.2.14 Inter-Application Communication policy.

When a signed application is granted `OcapIxcPermission` for "lookup" with a specified scope of "service-\*", that application SHALL be able to lookup objects bound by applications in services other than its own.

The installed `SecurityPolicyHandler`, if there is one, MAY deny these default permissions outright or grant only a subset of these permissions (see 21.2.1.9).

#### 14.2.2.9.3 Permission Syntax

```
<!ELEMENT ocap:ixc EMPTY>
<!ATTLIST ocap:ixc
  scope CDATA #IMPLIED
  old CDATA #IMPLIED
  aid CDATA #IMPLIED
  name CDATA #IMPLIED
  action (bind | lookup) #REQUIRED
```
The following table describes the attributes for an `<ocap:ixc/>` permission request file entry.

**Table 14–2 - Attributes for an `<ocap:ixc/>` permission request file entry**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Valid Values</th>
<th>Implied Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>scope</td>
<td>Specifies the scope.</td>
<td>&quot;service&quot; or &quot;xservice&quot;&lt;br&gt;Where &quot;service&quot; implies &quot;service-id&quot; and &quot;xservice&quot; implies &quot;service-*&quot;</td>
<td>&quot;service&quot;</td>
</tr>
<tr>
<td>oid</td>
<td>Organization Id.</td>
<td>*** or &quot;organization_id&quot;&lt;br&gt;Ignored for action=&quot;bind&quot;</td>
<td>*** for action=&quot;lookup&quot;&lt;br&gt;&quot;organization_id&quot; for action=&quot;bind&quot;</td>
</tr>
<tr>
<td>aid</td>
<td>Application Id.</td>
<td>*** or &quot;application_id&quot;&lt;br&gt;Ignored for action=&quot;bind&quot;</td>
<td>*** for action=&quot;lookup&quot;&lt;br&gt;&quot;application_id&quot; for action=&quot;bind&quot;</td>
</tr>
<tr>
<td>name</td>
<td>Bind name as specified on <code>IxcRegistry.bind()</code></td>
<td>*** or any arbitrary string</td>
<td>***</td>
</tr>
<tr>
<td>action</td>
<td>IxcRegistry action.</td>
<td>&quot;bind&quot; or &quot;lookup&quot;</td>
<td>n/a</td>
</tr>
</tbody>
</table>

If a valid permission request file contains one or more valid ocap:ixc entries, then the OcapIxcPermissions requested by the application SHALL be considered to be limited to those explicitly requested by the permission request file. The following example demonstrates the entry that would request the default privileges were no entry provided:

```xml
<ocap:ixc action="bind"/>
<ocap:ixc action="lookup"/>
```
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-MHP 1.0.3] model expects three planes:

- background plane
- video plane
- graphics plane

OCAP follows this reference model. Using this reference model, an OCAP 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-MHP 1.0.3] and [DVB-GEM 1.0.2] as follows:

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>MHP Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.1 DVB-GEM and DVB-MHP Specification Correspondence</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>15.2 OCAP Specific Requirements</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>15.2.1 Deviations from the DVB-MHP Specification</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>15.2.1.1 Introduction</td>
<td>13.0 General</td>
<td>Extension</td>
<td>13.1 Introduction</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>13.0 General</td>
<td>Compliance</td>
<td>13.1.1 Interapplication interaction</td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>13.0 General</td>
<td>Compliance</td>
<td>13.2 General Issues</td>
<td>A subsection is extended</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>13.0 General</td>
<td>Compliance</td>
<td>13.2.1 Coordinate Spaces</td>
<td>A subsection is extended</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>13.0 General</td>
<td>Compliance</td>
<td>13.2.1.1 Normalized screen space</td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>13.0 General</td>
<td>Compliance</td>
<td>13.2.1.2 User space</td>
<td>Compliance</td>
</tr>
</tbody>
</table>
15.2 OCAP Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

15.2.1 Deviations from the DVB-MHP Specification

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

15.2.1.1 Introduction

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 13.0 General and contains extensions of [DVB-MHP 1.0.3] Section: 13.1 Introduction.

In the OCAP graphics reference model the background plane is a logical construct, and may not necessarily be implemented in hardware.

OCAP does not support DVB subtitles. An OCAP implementation is not required to provide the DVB-MHP subtitle API or behavior discussed in this section of the [DVB-MHP 1.0.3].
15.2.1.2 Typical Resolutions

This subsection is compliant with [DVB-GEM 1.0.2] Section: 13.1 Supported graphics resolution and contains extensions of [DVB-MHP 1.0.3] 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-MHP 1.0.3]. The supported resolutions for OCAP are defined in Section 25.

<table>
<thead>
<tr>
<th>Resolutions for full-screen HGraphicsDevice</th>
<th>4:3 Display Pixel Aspect Ratio</th>
<th>16:9 Display Pixel Aspect Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>640 x 480</td>
<td>1:1</td>
<td>4:3</td>
</tr>
<tr>
<td>960x540</td>
<td>3:4</td>
<td>1:1</td>
</tr>
</tbody>
</table>

15.2.1.3 Composition of AWT Containers and Components

This subsection is compliant with [DVB-GEM 1.0.2] Section: 13.1 Supported graphics resolution and contains extensions of [DVB-MHP 1.0.3] Section: 13.3.4 Composition.

This section is an informative description for Section 13.0 of the [DVB-GEM 1.0.2].

If the AWT components are implemented as the heavyweight components, the discussion in Section 13.3 of [DVB-MHP 1.0.3] 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.

15.2.1.4 Modeling MPEG Decoding and Presentation Pipeline

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 13. Graphics Reference Model and contains extensions of [DVB-MHP 1.0.3] Section: 13.4.2 Modeling MPEG decoding and presentation pipeline.

References to DVB ETR 154 Standard Definition SHALL be replaced with references to [HOST].

15.2.1.5 Coordinate Spaces for Video

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 13. Graphics Reference Model and contains extensions of [DVB-MHP 1.0.3] Section: 13.4.3 Coordinate Spaces.

This section is compliant with Section 13.0 of the [DVB-GEM 1.0.2] except for the following modifications: References to [ETSI TR 101 154] Standard Definition SHALL be replaced with references to [HOST].

15.2.2 Extensions to DVB-GEM (Normative)

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

The following information is normative for OCAP. It is specific to OCAP and is not contained within the [DVB-MHP 1.0.3].
15.2.2.1 Closed-Captioning

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 13 Graphics reference model and contains extensions of [DVB-MHP 1.0.3] Section: 13 Graphics reference model.

OCAP 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 [CEA-608-E] and [CEA-708-D].

The closed captioning representation SHALL be drawn over any graphics of a Java application. The closed captioning layer is logically in the HgraphicsDevice. Actual implementation is manufacture dependent.

15.2.2.2 OCAP HScene Management

OCAP extends [DVB-GEM 1.0.2] 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 1.0.2] Section: 14 System integration aspects and contains extensions of [DVB-MHP 1.0.3] 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 1.0.2] Section. Section 16 (this section) of OCAP corresponds to Section 14 of [DVB-GEM 1.0.2] as follows:

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 System Integration Aspects</td>
<td>14 System integration aspects</td>
<td>Extension</td>
</tr>
<tr>
<td>16.1 DVB-GEM and DVB-MHP Specification Correspondence</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
<tr>
<td>16.2 OCAP Specific Requirements</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
<tr>
<td>16.2.1 Deviations from DVB-MHP</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
<tr>
<td>16.2.1.1 OCAP Locators</td>
<td>14.1 Namespace mapping</td>
<td>Extension</td>
</tr>
<tr>
<td>16.2.1.2 Reserved Names</td>
<td>14.2 Reserved names</td>
<td>Extension</td>
</tr>
<tr>
<td>16.2.1.3 XML notation</td>
<td>14.3 XML notation</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>14.4 Network signaling (error behaviour)</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>14.5 Text encoding of application identifiers</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>14.6 Filename requirements</td>
<td>Compliance</td>
</tr>
<tr>
<td>16.2.1.4 Files and File Names</td>
<td>14.7 Files and file names</td>
<td>Extension</td>
</tr>
<tr>
<td>16.2.1.5 Locators and content referencing</td>
<td>14.8 Locators and content referencing</td>
<td>Extension</td>
</tr>
<tr>
<td>16.2.1.6 Service Identification</td>
<td>14.9 Service identification</td>
<td>Compliance</td>
</tr>
<tr>
<td>16.2.1.7 CA System</td>
<td>14.10 CA system</td>
<td>Extension</td>
</tr>
<tr>
<td>16.2.1.8 Parental Control</td>
<td>No Corresponding Section</td>
<td>Extension</td>
</tr>
<tr>
<td>16.2.2 Extensions to DVB (normative)</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
</tbody>
</table>

16.2 OCAP Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

16.2.1 Deviations from DVB-MHP

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.
16.2.1.1 OCAP Locators

This section extends Section 14.1 of [DVB-GEM 1.0.2], 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. OCAP also provides support for referencing resources transmitted down the broadcast stream(s).

OCAP provides the OcapLocator and URLLocator instead of the DvbLocator defined in [DAVIC] and [DVB-MHP 1.0.3] Section 14.1. Applications SHALL use the OcapLocator or URLLocator.

16.2.1.1.1 DAVIC Locator Compliance Statement

The OcapLocator and URLLocator both extend the DAVIC Locator defined in [DAVIC] and are compliant with it and its URL format with a following modification.

OCAP uses the OCAP URL syntax for addressing the service and files within object carousels.

16.2.1.1.2 OCAP URL Naming Convention

Like [DVB-MHP 1.0.3], OCAP 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]

nocap://<source_id>[.<stream_type>[,<index>]][&<stream_type>[,<index>]][;<event_id>]/[/path_segments]

nocap://<source_id>[.+$<PID>{&<PID>}][;<event_id>]/[/path_segments]

nocap://n=<service_name>[.<stream_type>[,<ISO_639_language_code>]][&<stream_type>[,<ISO_639_language_code>]][;<event_id>]/[/path_segments]

nocap://n=<service_name>[.<stream_type>[,<index>]][&<stream_type>[,<index>]][;<event_id>]/[/path_segments]

nocap://n=<service_name>[.+<PID>{&<PID>}][;<event_id>]/[/path_segments]


nocap://f=<frequency>.<program_number> [.m=<modulation_format>] [.stream_type][.<index>] [;&<stream_type> [.stream_type][,<index>]][;<event_id>]/[/path_segments]

nocap://f=<frequency>.<program_number> [.m=<modulation_format>] [.stream_type][.+$<PID>{&<PID>}]
```

5/30/13
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
                    (See restriction 3 below)
index_elements    = stream_type [ "," index ] * ( "," index )
PID_elements      = "+" PID * ( "+" PID )
component_elements = "+" component_name * ( "+" component_name )
component_tag_elements = "+" component_tag * ( "+" component_tag )
ocap_program      = "+" frequency "." program_number [ ".m=" modulation_format ] |
                    "oobfdc." program_number
ocap_transport    = "+" 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 f below)
ISO_639_language_code = alpha alpha alpha
(See restriction g 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"

cap_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.
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.

stream_type shall conform to those defined in the "Stream type assignments" table of the [ISO 13818-1] specification and section 6.7 of [ATSC A/53-3].

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 1.0.2] 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 which is mandatorily present in all cable profiles.

<table>
<thead>
<tr>
<th>Name</th>
<th>Cable</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>source_id</td>
<td>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. 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.</td>
<td>Note that the specified stream is not guaranteed to be decoded if [HOST] does not support decoding it.</td>
</tr>
<tr>
<td>stream_type</td>
<td>The first program element matching that stream_type. The stream_types are defined in the stream_type assignments Table of [ISO 13818-1] and section 6.7 of [ATSC A/53-3]. Note that the specified stream is not guaranteed to be decoded if [HOST] does not support decoding it.</td>
<td></td>
</tr>
<tr>
<td>ISO 639 language code</td>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Cable</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>service_name</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>component_tag</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>component_name</td>
<td>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].</td>
<td></td>
</tr>
<tr>
<td>event_id</td>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>

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 Physical 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>
<thead>
<tr>
<th>Table 16–4 - OCAP URI Physical Layer Constructs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>frequency</td>
</tr>
<tr>
<td>oobfdbc</td>
</tr>
<tr>
<td>program_number</td>
</tr>
<tr>
<td>modulation_format</td>
</tr>
<tr>
<td>PID</td>
</tr>
<tr>
<td>Index in PMT</td>
</tr>
</tbody>
</table>

### 16.2.1.1.3 Examples of OCAP URL Usage

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 14.1 Namespace mapping.

The following Table shows example usage of OCAP locators.

<table>
<thead>
<tr>
<th>Table 16–5 - OCAP URL Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example</strong></td>
</tr>
<tr>
<td>ocap://0x1234</td>
</tr>
<tr>
<td>ocap://0x1234.0x2</td>
</tr>
<tr>
<td>ocap://0x1234.0x81</td>
</tr>
<tr>
<td>ocap://n=MOVIE</td>
</tr>
<tr>
<td>ocap://n=MOVIE.0x81,spa</td>
</tr>
<tr>
<td>ocap://0x1234/&lt;path&gt;/&lt;filename&gt;</td>
</tr>
<tr>
<td>ocap://0x1234.+0x56&amp;0x78;0xBC</td>
</tr>
<tr>
<td>ocap://f=0x23F10CC0.m=0x10</td>
</tr>
</tbody>
</table>
16.2.1.2 Reserved Names

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] 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 defined files.

16.2.1.3 XML notation

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] 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-MHP 1.0.3].

16.2.1.4 Files and File Names

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 14.7 Files and file names.

OCAP defines how applications can use the names of files in order to access content held in files. Like the [DVB-MHP 1.0.3], it is intentionally silent about the file systems and file system name-spaces of OCAP terminals except as defined below:

- When an OCAP application starts, the file system where that application is carried is mounted into the file system namespace for the OCAP terminal concerned. For an OCAP application, Section 11.5.1 [DVB-MHP 1.0.3] 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-MHP 1.0.3], Section 11.5.1.1, as modified by [DVB-GEM 1.0.2], 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 downloaded 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 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 applications MAY have the ability to mount additional file systems into the file system namespace of the OCAP 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 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 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 1.0.2] 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>
<thead>
<tr>
<th>Entity</th>
<th>Text Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport stream</td>
<td>OCAP locator including &quot;&lt;ocap_transport&gt;&quot; element</td>
</tr>
<tr>
<td>Network</td>
<td>No standardized text representation</td>
</tr>
<tr>
<td>OCAP Service</td>
<td>OCAP locator including &quot;&lt;ocap_service&gt;&quot; element</td>
</tr>
<tr>
<td>Generic Service</td>
<td>No standardized text representation unless also a OCAP Service</td>
</tr>
<tr>
<td>OCAP Event</td>
<td>OCAP locator including &quot;ocap_service_component&quot; element and &quot;event_id&quot; element.</td>
</tr>
<tr>
<td>MPEG Elementary Stream</td>
<td><code>ocap://&lt;ocap_service&gt;..&lt;program_element&gt;</code></td>
</tr>
</tbody>
</table>
| File                          | `file:` URL as defined in [RFC 1738]
|                               | `http:` URL as defined in [RFC 1738]
|                               | OCAP locator including "ocap_abs_path" element²             |
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 does not provide a textual representation for accessing this entity.

16.2.1.5.3 OCAP 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 an AlternativeContentErrorEvent with a reason code of CONTENT_NOT_FOUND; see section 13.3.9.3.

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 Implementation Constructors for OCAP Service Locators

When an application calls javax.tv.service.Service.getLocator(), the OCAP 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.

16.2.1.5.3.2 Non-unique Source Ids

Source IDs are considered unique by APIs that use source ID based locators to look up services, e.g., JavaTV. However, there are no network requirements for source IDs to be unique in a signaled VCT. In the event more than one service is signaled with the same source ID, the following implementation behaviors are defined:

- When multiple services are signaled with the same source ID, the implementation SHALL include all of those services in the JavaTV SI database.
- When any method is called and passes a parameter that contains a source ID term, e.g., SIManager.getService(Locator), ServiceContext.select(Locator []), and the source ID term identifies multiple services that are signaled simultaneously, the implementation SHALL use the identified service that was encountered first in the signaling.
- When the SIManager.filterServices(ServiceFilter) method is called, the implementation SHALL apply all services to the filter parameter, including those with identical source ID based locators. When the parameter is null, the implementation SHALL return all services, including those with identical source ID based locators.
- When the ServiceContext.select(Service) method is called, the implementation SHALL select the signaled service associated with the Service parameter.

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 Service is also valid as a Generic Service locator.

16.2.1.5.5 OCAP Event

This entity, represented by a locator, includes the "event_id" elements. OCAP 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 remote HTTP server via the interaction channel or on a currently mounted file system and can vary between OCAP 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 remote HTTP server via the interaction channel or on a currently mounted file system and can vary between OCAP 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-MHP 1.0.3].

16.2.1.6 Service Identification

This subsection complies with [DVB-GEM 1.0.2] Section 14.9 and defines a standardized textual representation for transport independent locators. The following assertions are made for transport independent and transport dependent locators.

- A transport independent locator SHALL be based on a source_id term.
- A transport dependent locator SHALL be based on a source_id, service_name, or frequency.program_number term.
- All service objects in the SI database returned the from javax.tv.service.SIManager SHALL be transport dependent.

In addition, this section extends [DVB-GEM 1.0.2] and defines an actual locator format used in the [DVB-MHP 1.1.3] provider SPI. The format of these locators SHALL contain a frequency and program_number term in order to be considered properly formatted.

16.2.1.7 CA System

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 14.10 CA system.

This section complies with and extends Section 14.10 of [DVB-GEM 1.0.2] in that both GEM and OCAP 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].

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] 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.

When selected video is not descrambled, the implementation SHALL black out the video in implementation-specific fashion. When selected audio is not descrambled, the implementation SHALL silence audio output in an implementation-specific fashion. This video and audio substitution is considered alternative content and causes events to be generated as described in this section.

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:
When the `ServiceContext(Locator [])` method is used, presentation can occur even if one or more requested service components cannot be presented; see [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.

![Flowchart](image)

*Figure 16–1 - CA event generation for ServiceContext.select(Service)*
ca_pmt_reply received

Any ES have a ca_enable value of 0x01?

Y

N

Any ES have a ca_enable value of 0x02 or 0x03?

Y

N

Any ES have a ca_enable value of 0x71 or 0x73?

Y

N

Generate events for tested ca_enable value as per table 16-7

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; see Figure 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 [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.
Table 16-7 - CA Events

<table>
<thead>
<tr>
<th>ca_enable field value returned in a ca_pmt_reply</th>
<th>ServiceContext Event generated</th>
<th>ServiceMediaHandler or discrete Player Event generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01 Descrambling possible.</td>
<td>NormalContentEvent</td>
<td>NormalMediaPresentationEvent</td>
</tr>
<tr>
<td>0x02 Descrambling possible under purchase conditions.</td>
<td>AlternativeContentErrorEvent with reason CA_REFUSAL.</td>
<td>AlternativeMediaPresentationEvent with reason NO_ENTITLEMENT.</td>
</tr>
<tr>
<td>0x03 Descrambling possible under technical conditions.</td>
<td>AlternativeContentErrorEvent with reason CA_REFUSAL.</td>
<td>AlternativeMediaPresentationEvent with reason NO_ENTITLEMENT.</td>
</tr>
<tr>
<td>0x71 Descrambling not possible, no purchase possible.</td>
<td>AlternativeContentErrorEvent with reason CA_REFUSAL.</td>
<td>AlternativeMediaPresentationEvent with reason NO_ENTITLEMENT.</td>
</tr>
<tr>
<td>0x73 Descrambling not possible, no resources available.</td>
<td>AlternativeContentErrorEvent with reason CA_REFUSAL.</td>
<td>AlternativeMediaPresentationEvent with reason HARDWARE_RESOURCE_NOT_AVAIL_ABLE.</td>
</tr>
</tbody>
</table>

NOTE: In case that 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.

16.2.1.8 Parental Control

This subsection contains requirements that do not correspond to any section in [DVB-GEM 1.0.2].

Parental control in OCAP is based on VBI signaling, content_advisory_descriptor, CCIF feature parameters, and responses from a registered MediaAccessHandler; see Section 21.2.1.22. Conditions for conditional access SHALL be determined before conditions for parental control. Event generation is determined for conditional access, and when conditional access is entitled (i.e., will generate a normal content event as per Section 16.2.1.7), event generation MAY be altered by parental control consideration. When service components in a presenting service context are newly-selected or change for any reason and conditional access is granted, and the change causes presentation to be blocked for parental control purposes, the implementation SHALL black out the video component and disable the audio component in an implementation-specific fashion. Circumstances where content is blocked for parental control purposes also include implementation responses and default behaviors to settings from the media access handler API; see Annex S. In addition, implementation responses to parental control in a feature parameter APDU MAY cause content blocking; see [CCIF].

Video replacement for parental control blocking is considered alternative content, and the implementation SHALL generate an org.ocap.service.AlternativeContentErrorEvent for any service context presenting blocked service components. The reason in the event SHALL be org.ocap.service.AlternativeContentErrorEvent.RATING_PROBLEM. In addition, the implementation SHALL generate an org.ocap.media.AlternativeMediaPresentationEvent for any JMF player presenting a service component blocked by a parental control. The reason in the event SHALL be org.ocap.media.AlternativeMediaPresentationReasons.RATING_PROBLEM. In this case, the service context or player SHALL retain resources when allowed by resource contention handling as defined in Section 19. When any service component in a presenting service context is blocked, applications bound to the selected service SHALL NOT be allowed to run. Newly-signalized applications in an AIT SHALL NOT be launched, and running applications signaled from a previous service and included in the current service AIT SHALL be destroyed. When a blocked service is presented as alternative content and is unblocked due to change in parental control status, the implementation SHALL present the service and generate a javax.tv.service.selection.NormalContentEvent.

When conditional access is entitled and parental controls do not block presentation, the implementation SHALL generate a javax.tv.service.selection.NormalContentEvent in accordance with [Java TV] and Section 16.2.1.7. In addition, an org.ocap.media.NormalMediaPresentationEvent is generated for JMF players that are CA entitled and are not blocked by parental controls, also in accordance with Section 16.2.1.7.
When both a PMT and an Aggregate Event Information Table (AEIT, defined in [SCTE 65]) contain a content_advisory_descriptor for the same service and the rating information does not match, the implementation SHALL use the rating information in the AEIT.

16.2.2 Extensions to DVB (normative)

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] 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 1.0.2] Section: 15 Detailed platform profile definitions and are extensions of [DVB-MHP 1.0.3] 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-MHP 1.0.3]). OCAP 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 1.0.2] Section.

Section 17 (this section) of OCAP corresponds to Section 15 of [DVB-MHP 1.0.3] as follows:

| Table 17–1 - Correlation between OCAP and [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3] |
|---|---|---|---|---|
| OCAP | [DVB-GEM 1.0.2] Section | GEM Compliance | [DVB-MHP 1.0.3] 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 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 | | |

17.2 OCAP Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.
17.2.1 Deviations from DVB-MHP

This subsection contains OCAP-specific requirements that does not correspond to any [DVB-GEM 1.0.2] Section.

Section 17 (this section) of OCAP deviates from Section 15.2 of [DVB-MHP 1.0.3] as follows:

17.2.1.1 Minimum Platform Profile

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 15.2 Minimum media formats supported by DVB-J APIs and are extensions of [DVB-MHP 1.0.3] Section: 15.2 Minimum media formats supported by DVB-J APIs.

OCAP deviates from Section 15.2 of [DVB-MHP 1.0.3] as follows:

Support for the Interactive Profile 1 in Section 16, Table 6 of [DVB-GEM 1.0.2] (Section 15, Table 65 of [DVB-MHP 1.0.3]) is required for OCAP, 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 1.0.2] Section: 15.4 Locale support and are extensions of [DVB-MHP 1.0.3] Section: 15.4 Locale support.

OCAP extends Section 15.4 of [DVB-GEM 1.0.2] as follows:

Support for the locale EN and EN.US is required. The default SHALL be EN.US.
18 REGISTRY OF CONSTANTS

This section contains requirements that are extensions to [DVB-GEM 1.0.2] Section: 16 Registry of constants and are extensions of [DVB-MHP 1.0.3] 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 1.0.2] Section.

Section 18 (this section) of OCAP corresponds to Section 16 of [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2] as follows:

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>MHP Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>18 Registry of Constants</td>
<td>16 Registry of constants</td>
<td>Extension</td>
<td>16 Registry of Constants</td>
<td>Extension</td>
</tr>
<tr>
<td>18.1 DVB-GEM and DVB-MHP Specification Correspondence</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>18.2 OCAP Specific Requirements</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>18.2.1 Deviations from the DVB-MHP Specification</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>18.2.1.1 System Constants</td>
<td>16.1 System constants</td>
<td>Compliance</td>
<td>Table 69, Profile encoding</td>
<td>Extension</td>
</tr>
<tr>
<td>18.2.1.2 JAVA Constants</td>
<td>16.2 DVB-J constants</td>
<td>Deviation</td>
<td>16.2 DVB-J constants</td>
<td>Extension</td>
</tr>
<tr>
<td>18.2.1.3 OCAP-Specific JAVA Constants</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
</tbody>
</table>

18.2 OCAP Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

18.2.1 Deviations from the DVB-MHP Specification

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

18.2.1.1 System Constants

This subsection is compliant with [DVB-GEM 1.0.2] Section: 16.1 System constants and are extensions of [DVB-MHP 1.0.3] Section: 16.1 System constants.

The profile encoding definitions in Table 69 of [DVB-MHP 1.0.3]: Profile encoding SHALL be considered extended as follows:
Table 18–2 - OCAP Extension to Table 69 of [DVB-MHP 1.0.3]

<table>
<thead>
<tr>
<th>application profile</th>
<th>major</th>
<th>minor</th>
<th>micro</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x102</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>As used in extending Table 69 of [DVB-MHP 1.0.3], the application profile will always be defined as 0x102 and the version will be the OCAP Bundle release version.</td>
</tr>
</tbody>
</table>

18.2.1.2  JAVA Constants

This subsection is compliant with [DVB-GEM 1.0.2] Section: 16.2 DVB-J constants and contains extensions of [DVB-MHP 1.0.3] Section: 16.2 DVB-J constants.

OCAP complies with Section 16.2 of [DVB-GEM 1.0.2], except for the following deviation.

The constant value definitions for the following classes SHALL be as defined in [JSR 217] instead of [DVB-GEM 1.0.2] Section: 16.2 DVB-J constants:

```
java.awt.event.MouseEvent.MOUSE_LAST
java.awt.event.WindowEvent.WINDOW_LAST
java.net.HttpURLConnection.HTTP_INTERNAL_ERROR
```

18.2.1.3  OCAP-Specific JAVA Constants

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

All constants are public final. These OCAP-specific JAVA constants are set in the following packages:

```
org.ocap.application.AppPattern
```

<table>
<thead>
<tr>
<th>public static final int</th>
<th>ALLOW</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final int</td>
<td>ASK</td>
<td>3</td>
</tr>
<tr>
<td>public static final int</td>
<td>DENY</td>
<td>2</td>
</tr>
</tbody>
</table>

```
org.ocap.application.OcapAppAttributes
```

<table>
<thead>
<tr>
<th>public static final int</th>
<th>AUTOSTART</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final int</td>
<td>BACKGROUND_MODE</td>
<td>3</td>
</tr>
<tr>
<td>public static final int</td>
<td>CROSSENVIRONMENT_MODE</td>
<td>2</td>
</tr>
<tr>
<td>public static final int</td>
<td>DESTROY</td>
<td>3</td>
</tr>
<tr>
<td>public static final int</td>
<td>KILL</td>
<td>4</td>
</tr>
<tr>
<td>public static final int</td>
<td>LEGACY_MODE</td>
<td>0</td>
</tr>
<tr>
<td>public static final int</td>
<td>NORMAL_MODE</td>
<td>1</td>
</tr>
<tr>
<td>public static final int</td>
<td>OCAP_J</td>
<td>1</td>
</tr>
<tr>
<td>public static final int</td>
<td>PAUSED_MODE</td>
<td>4</td>
</tr>
<tr>
<td>public static final int</td>
<td>PREFETCH*</td>
<td>5</td>
</tr>
<tr>
<td>public static final int</td>
<td>PRESENT</td>
<td>2</td>
</tr>
<tr>
<td>public static final int</td>
<td>REMOTE</td>
<td>6</td>
</tr>
</tbody>
</table>

*Used only for DVB-HTML applications
<table>
<thead>
<tr>
<th>Class</th>
<th>Field/Resource</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.ocap.diagnostics.MIBDefinition</td>
<td>SNMP_TYPE_BITS</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>SNMP_TYPE_COUNTER32</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>SNMP_TYPE_COUNTER64</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>SNMP_TYPE_GAUGE32</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>SNMP_TYPE_INTEGER</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>SNMP_TYPE_INVALID</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>SNMP_TYPE_IPADDRESS</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>SNMP_TYPE_OBJECTID</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>SNMP_TYPE_OCTETSTRING</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>SNMP_TYPE_OPAQUE</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>SNMP_TYPE_TIMETICKS</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>ECM_SUBDEVICE</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>ESTB_SUBDEVICE</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>MIB_ACCESS_READONLY</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>MIB_ACCESS_READWRITE</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>MIB_ACCESS_WRITEONLY</td>
<td>2</td>
</tr>
<tr>
<td>org.ocap.diagnostics.SNMPRequest</td>
<td>SNMP_CHECK_FOR_SET_REQUEST</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>SNMP_GET_NEXT_REQUEST</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>SNMP_GET_REQUEST</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>SNMP_SET_REQUEST</td>
<td>1</td>
</tr>
<tr>
<td>org.ocap.diagnostics.SNMPResponse</td>
<td>SNMP_REQUEST_AUTHORIZATION_ERROR</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>SNMP_REQUEST_BAD_VALUE</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>SNMP_REQUEST_COMMIT_FAILED</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>SNMP_REQUEST_GENERIC_ERROR</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>SNMP_REQUEST_INCONSISTENT_NAME</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>SNMP_REQUEST_INCONSISTENT_VALUE</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>SNMP_REQUEST_NO_ACCESS</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>SNMP_REQUEST_NO_CREATION</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>SNMP_REQUEST_NO_SUCH_NAME</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>SNMP_REQUEST_NOT_WRITABLE</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>SNMP_REQUEST_READ_ONLY</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>SNMP_REQUEST_RESOURCE_UNAVAILABLE</td>
<td>13</td>
</tr>
<tr>
<td>Class</td>
<td>Field</td>
<td>Value</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>--------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>org.ocap.diagnostics.SNMPResponse</td>
<td>SNMP_REQUEST_SUCCESS</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>SNMP_REQUEST_TOO_BIG</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>SNMP_REQUEST_UNDO_FAILED</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>SNMP_REQUEST_WRONG_ENCODING</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>SNMP_REQUEST_WRONG_LENGTH</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>SNMP_REQUEST_WRONG_TYPE</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>SNMP_REQUEST_WRONG_VALUE</td>
<td>10</td>
</tr>
<tr>
<td>org.ocap.hardware.Host</td>
<td>FULL_POWER</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>LOW_POWER</td>
<td>2</td>
</tr>
<tr>
<td>org.ocap.hardware.VideoOutputPort</td>
<td>AV_OUTPUT_PORT_TYPE_1394</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>AV_OUTPUT_PORT_TYPE_BB</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>AV_OUTPUT_PORT_TYPE_COMPONENT_VIDEO</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>AV_OUTPUT_PORT_TYPE_DVI</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>AV_OUTPUT_PORT_TYPE_HDMI</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>AV_OUTPUT_PORT_TYPE_INTERNAL</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>AV_OUTPUT_PORT_TYPE_RF</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>AV_OUTPUT_PORT_TYPE_SVIDEO</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>CAPABILITY_TYPE_DTCP</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CAPABILITY_TYPE_HDCP</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>CAPABILITY_TYPE_RESOLUTION_RESTRICTION</td>
<td>2</td>
</tr>
<tr>
<td>org.ocap.hardware.pod.PODApplication</td>
<td>TYPE_CA</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>TYPE_CP</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>TYPE_DIAGNOSTIC</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>TYPE_DVS167</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>TYPE_DVS178</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>TYPE_IP</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>TYPE_UNDESIGNATED</td>
<td>7</td>
</tr>
<tr>
<td>org.ocap.media.AlternativeMediaPresentationReason</td>
<td>BROADCAST_INCONSISTENCY</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>CA_UNKNOWN</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>COMMERCIAL_DIALOG</td>
<td>2</td>
</tr>
</tbody>
</table>
### org.ocap.media.AlternativeMediaPresentationReason

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final int</td>
<td>HARDWARE_RESOURCE_NOTAVAILABLE</td>
<td>32</td>
</tr>
<tr>
<td>public static final int</td>
<td>NO_ENTITLEMENT</td>
<td>1</td>
</tr>
<tr>
<td>public static final int</td>
<td>RATING_PROBLEM</td>
<td>4</td>
</tr>
<tr>
<td>public static final int</td>
<td>REASON_FIRST</td>
<td>1</td>
</tr>
<tr>
<td>public static final int</td>
<td>REASON_LAST</td>
<td>32</td>
</tr>
</tbody>
</table>

### org.ocap.media.ClosedCaptioningAttribute

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final int</td>
<td>CC_ATTRIBUTE_FONT_ITALICIZED</td>
<td>6</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_ATTRIBUTE_FONT_STYLE</td>
<td>4</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_ATTRIBUTE_FONT_UNDERLINE</td>
<td>7</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_ATTRIBUTE_PEN_BG_COLOR</td>
<td>1</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_ATTRIBUTE_PEN_BG_OPACITY</td>
<td>3</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_ATTRIBUTE_PEN_FG_COLOR</td>
<td>0</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_ATTRIBUTE_PEN_FG_OPACITY</td>
<td>2</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_ATTRIBUTE_PEN_SIZE</td>
<td>5</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_ATTRIBUTE_WINDOW_BORDER_COLOR</td>
<td>11</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_ATTRIBUTE_WINDOW_BORDER_TYPE</td>
<td>10</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_ATTRIBUTE_WINDOW_FILL_COLOR</td>
<td>8</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_ATTRIBUTE_WINDOW_FILL_OPACITY</td>
<td>9</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_BORDER_DEPRESSED</td>
<td>2</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_BORDER_NONE</td>
<td>0</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_BORDER_RAISED</td>
<td>1</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_BORDER_SHADOW_LEFT</td>
<td>4</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_BORDER_SHADOW_RIGHT</td>
<td>5</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_BORDER_UNIFORM</td>
<td>3</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_OPACITY_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_OPACITY_SOLID</td>
<td>0</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_OPACITY_TRANSLUCENT</td>
<td>2</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_OPACITY_TRANSPARENT</td>
<td>3</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_PEN_SIZE_LARGE</td>
<td>2</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_PEN_SIZE_SMALL</td>
<td>0</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_PEN_SIZE_STANDARD</td>
<td>1</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_TYPE_ANALOG</td>
<td>0</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_TYPE_DIGITAL</td>
<td>1</td>
</tr>
</tbody>
</table>

### org.ocap.media.ClosedCaptioningControl

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final int</td>
<td>CC_ANALOG_SERVICE_CC1</td>
<td>1000</td>
</tr>
<tr>
<td>public static final int</td>
<td>CC_ANALOG_SERVICE_CC2</td>
<td>1001</td>
</tr>
<tr>
<td>Class</td>
<td>Member</td>
<td>Value</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>---------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>org.ocap.media.ClosedCaptioningControl</td>
<td>public static final int CC_ANALOG_SERVICE_CC3</td>
<td>1002</td>
</tr>
<tr>
<td></td>
<td>public static final int CC_ANALOG_SERVICE_CC4</td>
<td>1003</td>
</tr>
<tr>
<td></td>
<td>public static final int CC_ANALOG_SERVICE_T1</td>
<td>1004</td>
</tr>
<tr>
<td></td>
<td>public static final int CC_ANALOG_SERVICE_T2</td>
<td>1005</td>
</tr>
<tr>
<td></td>
<td>public static final int CC_ANALOG_SERVICE_T3</td>
<td>1006</td>
</tr>
<tr>
<td></td>
<td>public static final int CC_ANALOG_SERVICE_T4</td>
<td>1007</td>
</tr>
<tr>
<td></td>
<td>public static final int CC_NO_SERVICE</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>public static final int CC_TURN_OFF</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>public static final int CC_TURN_ON</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>public static final int CC_TURN_ON_MUTE</td>
<td>2</td>
</tr>
<tr>
<td>org.ocap.media.ClosedCaptioningEvent</td>
<td>public static final int EVENTID_CLOSED_CAPTIONING_OFF</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>public static final int EVENTID_CLOSED_CAPTIONING_ON</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>public static final int EVENTID_CLOSED_CAPTIONING_ON_MUTE</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>public static final int EVENTID_CLOSED_CAPTIONING_SELECT_NEW_SERVICE</td>
<td>3</td>
</tr>
<tr>
<td>org.ocap.media.MediaTimerListener</td>
<td>public static final int TIMER_START</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>public static final int TIMER_STOP</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>public static final int TIMER_WENTOFF_FIRST</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>public static final int TIMER_WENTOFF_LAST</td>
<td>4</td>
</tr>
<tr>
<td>org.ocap.media.S3DConfiguration</td>
<td>public static final int S3D_AVC_SEI_PAYLOAD_TYPE</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>public static final int S3D_MPEG2_USER_DATA_TYPE</td>
<td>2</td>
</tr>
<tr>
<td>org.ocap.media.S3DFormatTypes</td>
<td>public static final int FORMAT_2D</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>public static final int SIDE_BY_SIDE</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>public static final int TOP_AND_BOTTOM</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>public static final int UNKNOWN</td>
<td>0</td>
</tr>
<tr>
<td>org.ocap.media.S3DSignalingChangedEvent</td>
<td>public static final int TRANSITION_FROM_2D_TO_3D</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>public static final int TRANSITION_FROM_3D_TO_2D</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>public static final int TRANSITION_OF_3D_FORMAT</td>
<td>3</td>
</tr>
</tbody>
</table>
### org.ocap.media.VBIFilterEvent

<table>
<thead>
<tr>
<th>Method</th>
<th>Event Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVENT_CODE_BUFFER_FULL</td>
<td>public static final int</td>
<td>6</td>
</tr>
<tr>
<td>EVENT_CODE_FAILED_TO_DESCRAMBLE</td>
<td>public static final int</td>
<td>4</td>
</tr>
<tr>
<td>EVENT_CODE_FIRST_VBI_DATA_AVAILABLE</td>
<td>public static final int</td>
<td>1</td>
</tr>
<tr>
<td>EVENT_CODE_FORCIBLE_TERMINATED</td>
<td>public static final int</td>
<td>2</td>
</tr>
<tr>
<td>EVENT_CODE_TIME_NOTIFICATION</td>
<td>public static final int</td>
<td>7</td>
</tr>
<tr>
<td>EVENT_CODE_UNITS_NOTIFICATION</td>
<td>public static final int</td>
<td>8</td>
</tr>
<tr>
<td>EVENT_CODE_VIDEO_SOURCE_CHANGED</td>
<td>public static final int</td>
<td>3</td>
</tr>
</tbody>
</table>

### org.ocap.media.VBIFilterGroup

<table>
<thead>
<tr>
<th>Method</th>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIELD_1</td>
<td>public static final int</td>
<td>1</td>
</tr>
<tr>
<td>FIELD_2</td>
<td>public static final int</td>
<td>2</td>
</tr>
<tr>
<td>FIELD_MIXED</td>
<td>public static final int</td>
<td>3</td>
</tr>
<tr>
<td>VBI_DATA_FORMAT_AMOL_I</td>
<td>public static final int</td>
<td>8</td>
</tr>
<tr>
<td>VBI_DATA_FORMAT_AMOL_II</td>
<td>public static final int</td>
<td>9</td>
</tr>
<tr>
<td>VBI_DATA_FORMAT_EIA608B</td>
<td>public static final int</td>
<td>6</td>
</tr>
<tr>
<td>VBI_DATA_FORMAT_NABTS</td>
<td>public static final int</td>
<td>7</td>
</tr>
<tr>
<td>VBI_DATA_FORMAT_T1</td>
<td>public static final int</td>
<td>2</td>
</tr>
<tr>
<td>VBI_DATA_FORMAT_T2</td>
<td>public static final int</td>
<td>3</td>
</tr>
<tr>
<td>VBI_DATA_FORMAT_T3</td>
<td>public static final int</td>
<td>4</td>
</tr>
<tr>
<td>VBI_DATA_FORMAT_T4</td>
<td>public static final int</td>
<td>5</td>
</tr>
<tr>
<td>VBI_DATA_FORMAT_UNKNOWN</td>
<td>public static final int</td>
<td>10</td>
</tr>
<tr>
<td>VBI_DATA_FORMAT_XDS</td>
<td>public static final int</td>
<td>1</td>
</tr>
</tbody>
</table>

### org.ocap.media.VideoFormatControl

<table>
<thead>
<tr>
<th>Method</th>
<th>Scanmode</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCANMODE_INTERLACED</td>
<td>public static final int</td>
<td>1</td>
</tr>
<tr>
<td>SCANMODE_PROGRESSIVE</td>
<td>public static final int</td>
<td>2</td>
</tr>
<tr>
<td>SCANMODE_UNKNOWN</td>
<td>public static final int</td>
<td>0</td>
</tr>
</tbody>
</table>

### org.ocap.net.OCRCInterface

<table>
<thead>
<tr>
<th>Method</th>
<th>Subtype</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBTYPE_CATV_DOCSIS</td>
<td>public static final int</td>
<td>1</td>
</tr>
<tr>
<td>SUBTYPE_CATV_OOB</td>
<td>public static final int</td>
<td>2</td>
</tr>
</tbody>
</table>

### org.ocap.service.AlternativeContentErrorEvent

<table>
<thead>
<tr>
<th>Method</th>
<th>Error Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA_REFusal</td>
<td>public static final int</td>
<td>101</td>
</tr>
<tr>
<td>CONTENT_NOT_FOUND</td>
<td>public static final int</td>
<td>102</td>
</tr>
<tr>
<td>MISSING_HANDLER</td>
<td>public static final int</td>
<td>103</td>
</tr>
<tr>
<td>RATING_PROBLEM</td>
<td>public static final int</td>
<td>100</td>
</tr>
<tr>
<td>Class</td>
<td>Fields</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>org.ocap.service.AlternativeContentErrorEvent</code></td>
<td><code>public static final int TUNING_FAILURE 104</code></td>
<td></td>
</tr>
<tr>
<td><code>org.ocap.service.S3DAlternativeContentErrorEvent</code></td>
<td><code>public static final int S3D_FORMAT_NOT_SUPPORTED 106</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final int S3D_NO_HDMI_CONNECTION 107</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final int S3D_NOT_SUPPORTED 105</code></td>
<td></td>
</tr>
<tr>
<td><code>org.ocap.service.ServiceTypePermission</code></td>
<td><code>public static final java.lang.String BROADCAST &quot;broadcast&quot;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final java.lang.String MFR &quot;abstract.manufacturer&quot;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final java.lang.String MSO &quot;abstract.mso&quot;</code></td>
<td></td>
</tr>
<tr>
<td><code>org.ocap.si.DescriptorTag</code></td>
<td><code>public static final short AC3_AUDIO 129</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short APPLICATION 0</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short APPLICATION_ICONS 11</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short APPLICATION_NAME 1</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short APPLICATION_SIGNALING 111</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short ASSOCIATION_TAG 20</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short ATSC_PRIVATE_INFORMATION 173</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short AUDIO_STREAM 3</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short CA 9</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short CACHING_PRIORITY 113</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short CAPTION_SERVICE 134</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short CAROUSEL_IDENTIFIER 19</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short CHANNEL_PROPERTIES 149</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short COMPONENT_NAME 163</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short CONTENT_ADVISORY 135</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short CONTENT_TYPE 114</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short COPYRIGHT 13</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short DATA_BROADCAST_ID 102</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short DATA_STREAM_ALIGNMENT 6</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short DAYLIGHT_SAVINGS_TIME 150</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short DII_LOCATION 13</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short DVB_J_APPLICATION 3</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short DVB_J_APPLICATION_LOCATION 4</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short EXTENDED_CHANNEL_NAME_DESCRIPTION 160</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public static final short EXTERNAL_APPLICATION_AUTHORISATION 5</code></td>
<td></td>
</tr>
<tr>
<td>org.ocap.si.DescriptorTag</td>
<td>HIERARCHY</td>
<td>IBP</td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
<td>-----------</td>
<td>-----</td>
</tr>
<tr>
<td>public static final short</td>
<td>4</td>
<td>18</td>
</tr>
<tr>
<td>org.ocap.si.StreamType</td>
<td>AAC_ADTS_AUDIO</td>
<td>15</td>
</tr>
<tr>
<td>public static final short</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5/30/13

CableLabs®
### org.ocap.si.StreamType

<table>
<thead>
<tr>
<th>Method</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final short DSM_CC_STREAM_DESCRIPTORs</td>
<td>12</td>
<td>DSM_CC_STREAM_DESCRIPTORs</td>
</tr>
<tr>
<td>public static final short DSM_CC_UN</td>
<td>11</td>
<td>DSM_CC_UN</td>
</tr>
<tr>
<td>public static final short ENHANCED_ATSC_AUDIO</td>
<td>135</td>
<td>ENHANCED_ATSC_AUDIO</td>
</tr>
<tr>
<td>public static final short FLEXMUX_PES</td>
<td>18</td>
<td>FLEXMUX_PES</td>
</tr>
<tr>
<td>public static final short FLEXMUX_SECTIONS</td>
<td>19</td>
<td>FLEXMUX_SECTIONS</td>
</tr>
<tr>
<td>public static final short H_222</td>
<td>9</td>
<td>H_222</td>
</tr>
<tr>
<td>public static final short ISO_14496_VISUAL</td>
<td>16</td>
<td>ISO_14496_VISUAL</td>
</tr>
<tr>
<td>public static final short ISOCHRONOUS_DATA</td>
<td>131</td>
<td>ISOCHRONOUS_DATA</td>
</tr>
<tr>
<td>public static final short METADATA_DATA_CAROUSEL</td>
<td>23</td>
<td>METADATA_DATA_CAROUSEL</td>
</tr>
<tr>
<td>public static final short METADATA_OBJECT_CAROUSEL</td>
<td>24</td>
<td>METADATA_OBJECT_CAROUSEL</td>
</tr>
<tr>
<td>public static final short METADATA_PES</td>
<td>21</td>
<td>METADATA_PES</td>
</tr>
<tr>
<td>public static final short METADATA_SECTIONS</td>
<td>22</td>
<td>METADATA_SECTIONS</td>
</tr>
<tr>
<td>public static final short METADATA_SYNCH_DOWNLOAD</td>
<td>25</td>
<td>METADATA_SYNCH_DOWNLOAD</td>
</tr>
<tr>
<td>public static final short MHEG</td>
<td>7</td>
<td>MHEG</td>
</tr>
<tr>
<td>public static final short MPEG_1_AUDIO</td>
<td>3</td>
<td>MPEG_1_AUDIO</td>
</tr>
<tr>
<td>public static final short MPEG_1_VIDEO</td>
<td>1</td>
<td>MPEG_1_VIDEO</td>
</tr>
<tr>
<td>public static final short MPEG_2_AUDIO</td>
<td>4</td>
<td>MPEG_2_AUDIO</td>
</tr>
<tr>
<td>public static final short MPEG_2_VIDEO</td>
<td>2</td>
<td>MPEG_2_VIDEO</td>
</tr>
<tr>
<td>public static final short MPEG_PRIVATE_DATA</td>
<td>6</td>
<td>MPEG_PRIVATE_DATA</td>
</tr>
<tr>
<td>public static final short MPEG_PRIVATE_SECTION</td>
<td>5</td>
<td>MPEG_PRIVATE_SECTION</td>
</tr>
<tr>
<td>public static final short STD_SUBTITLE</td>
<td>130</td>
<td>STD_SUBTITLE</td>
</tr>
<tr>
<td>public static final short SYNCHRONIZED_DOWNLOAD</td>
<td>20</td>
<td>SYNCHRONIZED_DOWNLOAD</td>
</tr>
<tr>
<td>public static final short VIDEO_DCII</td>
<td>128</td>
<td>VIDEO_DCII</td>
</tr>
</tbody>
</table>

### org.ocap.storage.StorageManagerEvent

<table>
<thead>
<tr>
<th>Method</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final int STORAGE_PROXY_ADDED</td>
<td>0</td>
<td>STORAGE_PROXY_ADDED</td>
</tr>
<tr>
<td>public static final int STORAGE_PROXY_CHANGED</td>
<td>2</td>
<td>STORAGE_PROXY_CHANGED</td>
</tr>
<tr>
<td>public static final int STORAGE_PROXY_REMOVED</td>
<td>1</td>
<td>STORAGE_PROXY_REMOVED</td>
</tr>
</tbody>
</table>

### org.ocap.storage.StorageProxy

<table>
<thead>
<tr>
<th>Method</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final int BUSY</td>
<td>2</td>
<td>BUSY</td>
</tr>
<tr>
<td>public static final int DEVICE_ERROR</td>
<td>6</td>
<td>DEVICE_ERROR</td>
</tr>
<tr>
<td>public static final int NOT_PRESENT</td>
<td>7</td>
<td>NOT_PRESENT</td>
</tr>
<tr>
<td>public static final int OFFLINE</td>
<td>1</td>
<td>OFFLINE</td>
</tr>
<tr>
<td>public static final int READY</td>
<td>0</td>
<td>READY</td>
</tr>
<tr>
<td>public static final int UNINITIALIZED</td>
<td>5</td>
<td>UNINITIALIZED</td>
</tr>
<tr>
<td>public static final int UNSUPPORTED_DEVICE</td>
<td>3</td>
<td>UNSUPPORTED_DEVICE</td>
</tr>
</tbody>
</table>
org.ocap.storage.StorageProxy

```java
public static final int UNSUPPORTED_FORMAT = 4;
```

org.ocap.system.EASEvent

```java
public static final int EAS_COMPLETE = 3;
public static final int EAS_DETAILS_CHANNEL = 1;
public static final int EAS_TEXT_DISPLAY = 2;
```

org.ocap.system.EASManager

```java
public static final int EAS_MESSAGE_IN_PROGRESS_STATE = 1;
public static final int EAS_MESSAGE_RECEIVED_STATE = 0;
public static final int EAS_NOT_IN_PROGRESS_STATE = 2;
```

org.ocap.system.EASModuleRegistrar

```java
public static final int EAS_ATTRIBUTE_BACK_COLOR = 5;
public static final int EAS_ATTRIBUTE_BACK_OPACITY = 7;
public static final int EAS_ATTRIBUTE_FONT_COLOR = 1;
public static final int EAS_ATTRIBUTE_FONT_FACE = 3;
public static final int EAS_ATTRIBUTE_FONT_OPACITY = 6;
public static final int EAS_ATTRIBUTE_FONT_SIZE = 4;
public static final int EAS_ATTRIBUTE_FONT_STYLE = 2;
```

org.ocap.system.event.CableCARDResetEvent

```java
public static final int CABLECARD_RESET_BEGIN = 1493172224;
public static final int CABLECARD_RESET_COMPLETE = 1493172225;
```

org.ocap.system.event.ErrorEvent

```java
public static final int APP_CAT_GENERAL_ERROR = 1006632960;
public static final int APP_INFO_GENERAL_EVENT = 201326592;
public static final int APP_REC_GENERAL_ERROR = 738197504;
public static final int APP_REC_JAVA_THROWABLE = 738197505;
public static final int SYS_CAT_GENERAL_ERROR = 872415232;
public static final int SYS_CAT_JAVA_THROWABLE = 872415233;
public static final int SYS_INFO_GENERAL_EVENT = 67108864;
public static final int SYS_REC_GENERAL_ERROR = 603979776;
```

org.ocap.system.event.RebootEvent

```java
public static final int REBOOT_BY_IMPLEMENTATION = 1140850688;
public static final int REBOOT_BY_TRUSTED_APP = 1140850691;
```
<table>
<thead>
<tr>
<th>Package</th>
<th>Event</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.ocap.system.event.RebootEvent</td>
<td>REBOOT_FOR_UNRECOVERABLE_HW_ERROR</td>
<td>1140850690</td>
</tr>
<tr>
<td></td>
<td>REBOOT_FOR_UNRECOVERABLE_SYS_ERROR</td>
<td>1140850689</td>
</tr>
<tr>
<td>org.ocap.system.event.ResourceDepletionEvent</td>
<td>RESOURCE_CPU_BANDWIDTH_DEPLETED</td>
<td>1409286146</td>
</tr>
<tr>
<td></td>
<td>RESOURCE_RC_BANDWIDTH_DEPLETED</td>
<td>1409286147</td>
</tr>
<tr>
<td></td>
<td>RESOURCE_SYS_MEM_DEPLETED</td>
<td>1409286144</td>
</tr>
<tr>
<td></td>
<td>RESOURCE_VM_MEM_DEPLETED</td>
<td>1409286145</td>
</tr>
<tr>
<td>org.ocap.system.event.SystemEvent</td>
<td>BEGIN_APP_CAT_ERROR_EVENT_TYPES</td>
<td>939524096</td>
</tr>
<tr>
<td></td>
<td>BEGIN_APP_CAT_ERROR_RESERVED_EVENT_TYPES</td>
<td>1006632960</td>
</tr>
<tr>
<td></td>
<td>BEGIN_APP_INFO_EVENT_TYPES</td>
<td>134217728</td>
</tr>
<tr>
<td></td>
<td>BEGIN_APP_INFO_RESERVED_EVENT_TYPES</td>
<td>201326592</td>
</tr>
<tr>
<td></td>
<td>BEGIN_APP_REC_ERROR_EVENT_TYPES</td>
<td>671088640</td>
</tr>
<tr>
<td></td>
<td>BEGIN_APP_REC_ERROR_RESERVED_EVENT_TYPES</td>
<td>738197504</td>
</tr>
<tr>
<td></td>
<td>BEGIN_SYS_CABLECARD_RESET_EVENT_TYPES</td>
<td>1493172224</td>
</tr>
<tr>
<td></td>
<td>BEGIN_SYS_CAT_ERROR_EVENT_TYPES</td>
<td>805306368</td>
</tr>
<tr>
<td></td>
<td>BEGIN_SYS_CAT_ERROR_RESERVED_EVENT_TYPES</td>
<td>872415232</td>
</tr>
<tr>
<td></td>
<td>BEGIN_SYS_DNLD_EVENT_TYPES</td>
<td>1476395008</td>
</tr>
<tr>
<td></td>
<td>BEGIN_SYS_INFO_EVENT_TYPES</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>BEGIN_SYS_INFO_RESERVED_EVENT_TYPES</td>
<td>67108864</td>
</tr>
<tr>
<td></td>
<td>BEGIN_SYS_REBOOT_EVENT_TYPES</td>
<td>1073741824</td>
</tr>
<tr>
<td></td>
<td>BEGIN_SYS_REBOOT_RESERVED_EVENT_TYPES</td>
<td>1140850688</td>
</tr>
<tr>
<td></td>
<td>BEGIN_SYS_REC_ERROR_EVENT_TYPES</td>
<td>536870912</td>
</tr>
<tr>
<td></td>
<td>BEGIN_SYS_REC_ERROR_RESERVED_EVENT_TYPES</td>
<td>603979776</td>
</tr>
<tr>
<td></td>
<td>BEGIN_SYS_RES_DEP_EVENT_TYPES</td>
<td>1342177280</td>
</tr>
<tr>
<td></td>
<td>BEGIN_SYS_RES_DEP_RESERVED_EVENT_TYPES</td>
<td>1409286144</td>
</tr>
<tr>
<td></td>
<td>END_APP_CAT_ERROR_EVENT_TYPES</td>
<td>1073741823</td>
</tr>
<tr>
<td></td>
<td>END_APP_INFO_EVENT_TYPES</td>
<td>536870911</td>
</tr>
<tr>
<td></td>
<td>END_APP_REC_ERROR_EVENT_TYPES</td>
<td>805306367</td>
</tr>
<tr>
<td></td>
<td>END_SYS_CABLECARD_RESET_EVENT_TYPES</td>
<td>1509949439</td>
</tr>
<tr>
<td></td>
<td>END_SYS_CAT_ERROR_EVENT_TYPES</td>
<td>939524095</td>
</tr>
<tr>
<td></td>
<td>END_SYS_DNLD_EVENT_TYPES</td>
<td>1493172223</td>
</tr>
<tr>
<td></td>
<td>END_SYS_REBOOT_EVENT_TYPES</td>
<td>134217727</td>
</tr>
<tr>
<td></td>
<td>END_SYS_REBOOT_RESERVED_EVENT_TYPES</td>
<td>1207959551</td>
</tr>
<tr>
<td></td>
<td>END_SYS_RES_DEP_EVENT_TYPES</td>
<td>671088639</td>
</tr>
</tbody>
</table>
### org.ocap.system.event.SystemEventManager

<table>
<thead>
<tr>
<th>Method</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final int CABLE_CARD_LISTENER</td>
<td>4</td>
</tr>
<tr>
<td>public static final int DEFERRED_DOWNLOAD</td>
<td>3</td>
</tr>
<tr>
<td>public static final int ERROR_LISTENER</td>
<td>0</td>
</tr>
<tr>
<td>public static final int REBOOT_LISTENER</td>
<td>1</td>
</tr>
<tr>
<td>public static final int RESOURCE_DEPLETION</td>
<td>2</td>
</tr>
</tbody>
</table>

### org.ocap.test.OCAPTest

<table>
<thead>
<tr>
<th>Method</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final int MAX_MESSAGE_LENGTH</td>
<td>1500</td>
</tr>
<tr>
<td>public static final byte MESSAGE_TERMINATION</td>
<td>0</td>
</tr>
<tr>
<td>public static final int TCP</td>
<td>1</td>
</tr>
<tr>
<td>public static final int UDP</td>
<td>0</td>
</tr>
</tbody>
</table>

### org.ocap.ui.event.OCRcEvent

<table>
<thead>
<tr>
<th>Method</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final int OCRC_FIRST</td>
<td>600</td>
</tr>
<tr>
<td>public static final int OCRC_LAST</td>
<td>630</td>
</tr>
<tr>
<td>public static final int VK_APPS</td>
<td>605</td>
</tr>
<tr>
<td>public static final int VK_BACK</td>
<td>608</td>
</tr>
<tr>
<td>public static final int VK_CC</td>
<td>630</td>
</tr>
<tr>
<td>public static final int VK_EXIT</td>
<td>601</td>
</tr>
<tr>
<td>public static final int VK_FORWARD</td>
<td>609</td>
</tr>
<tr>
<td>public static final int VK_INSTANT_REPLAY</td>
<td>627</td>
</tr>
<tr>
<td>public static final int VK_LAST</td>
<td>607</td>
</tr>
<tr>
<td>public static final int VK_LINK</td>
<td>606</td>
</tr>
<tr>
<td>public static final int VK_LIST</td>
<td>621</td>
</tr>
<tr>
<td>public static final int VK_LIVE</td>
<td>622</td>
</tr>
<tr>
<td>public static final int VK_LOCK</td>
<td>619</td>
</tr>
<tr>
<td>public static final int VK_MENU</td>
<td>602</td>
</tr>
<tr>
<td>public static final int VK_NEXT_DAY</td>
<td>603</td>
</tr>
<tr>
<td>public static final int VK_NEXT_FAVORITE_CHANNEL</td>
<td>612</td>
</tr>
<tr>
<td>public static final int VK_ON_DEMAND</td>
<td>623</td>
</tr>
<tr>
<td>public static final int VK_PINP_DOWN</td>
<td>626</td>
</tr>
<tr>
<td>public static final int VK_PINP_MOVE</td>
<td>624</td>
</tr>
<tr>
<td>public static final int VK_PINP_UP</td>
<td>625</td>
</tr>
<tr>
<td>public static final int VK_PREV_DAY</td>
<td>604</td>
</tr>
<tr>
<td>public static final int VK_RC_LOW_BATTERY</td>
<td>628</td>
</tr>
<tr>
<td>public static final int VK_RESERVE_1</td>
<td>613</td>
</tr>
<tr>
<td>public static final int VK_RESERVE_2</td>
<td>614</td>
</tr>
<tr>
<td></td>
<td>Method</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>org.ocap.ui.event.OCRcEvent</td>
<td>public static final int</td>
</tr>
<tr>
<td></td>
<td>public static final int</td>
</tr>
<tr>
<td></td>
<td>public static final int</td>
</tr>
<tr>
<td></td>
<td>public static final int</td>
</tr>
<tr>
<td></td>
<td>public static final int</td>
</tr>
<tr>
<td></td>
<td>public static final int</td>
</tr>
<tr>
<td></td>
<td>public static final int</td>
</tr>
<tr>
<td></td>
<td>public static final int</td>
</tr>
<tr>
<td></td>
<td>public static final int</td>
</tr>
</tbody>
</table>
19 RESOURCE MANAGEMENT

This section contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

In OCAP, 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 and DVB-MHP 1.0 that cause resource management to be handled slightly differently in the two environments:

- In [DVB-MHP 1.0.3], 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, however, allows for both service-bound and unbounded applications. In OCAP, 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 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 in particular, be executed, it is unlikely that the applications would cooperate with one another concerning the sharing of resources.

- In the OCAP 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 prefers to vest that authority in the Monitor Application rather than the application manager. Note that resource over-subscription is still possible in [DVB-MHP 1.0.3] 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-MHP 1.0.3].

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>19 Resource Management</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
</tbody>
</table>

19.2 OCAP Specific Requirements

This requirement extends the specification requirements about resource management made to the [DVB-MHP 1.0.3].

19.2.1 Normative

The following sections are normative for OCAP.

19.2.1.1 Resource Management in OCAP

The [DVB-GEM 1.0.2] 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-
MHP 1.0.3]. OCAP 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 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`
- `org.ocap.media.VBIFilterGroup`

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(ResourceUsage, ResourceUsage[])` method returns null or a `ResourceContentionHandler` has not been associated with the `ResourceContentionManager`, the inter application resource management process uses the application priority attribute (defined in Section 10.2.2.5) 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

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. If any resources were reserved by the implementation before the Monitor Application was launched, or if any resources were reserved by the implementation without being explicitly or implicitly requested by OCAP applications, the method `getAppID()` for the `ResourceUsage` corresponding to the resource allocation SHALL return null.

The implementation MAY support the sharing of a single implicitly reserved resource among multiple uses as specified in an OCAP extension specification. Where the shared use of a single implicitly reserved resource is to be represented as a `ResourceUsage`, it SHALL be represented by an instance of `org.ocap.resource.SharedResourceUsage` that encapsulates each individual use. The implementation SHALL use the highest application priority represented within the `SharedResourceUsage` object for resolving any resource contention. If none of the contained `ResourceUsage` objects have associated `AppID`S, then the priority of the `SharedResourceUsage` SHALL be considered to be 0.

19.2.1.1.5.1 HVideoDevice Reservation

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.

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 tuner is required to complete the selection, the implementation SHALL ensure that an attempt to reserve an `org.davic.net.tuning.NetworkInterface` instance is made on behalf of the application. If reservation cannot be ensured due to contention, then the selection will fail. This implicit reservation SHALL persist for the duration of the service presentation until replaced by another, lost to contention, the `ServiceContext` is stopped or destroyed, or a new service is selected that does not require tuner resources. The implicit reservation SHALL persist even after the application that caused the implicit reservation is destroyed.

The implementation MAY share implicitly reserved `org.davic.net.tuning.NetworkInterface` resources among multiple uses where specified by an OCAP extension specification. The implementation SHALL NOT share implicitly reserved `org.davic.net.tuning.NetworkInterface` resources among multiple `ServiceContext` uses unless otherwise required by an OCAP extension specification. That is, unless otherwise specified, a `SharedResourceUsage` will not be used.

The following behavior applies when a `ServiceContext` requires a tuner to present a service.
If the sharing of resources is otherwise required and an existing implicitly reserved NetworkInterface can satisfy the need, then:

- Verify that the application can reserve the resource per Section 19.2.1.1.1.
- If the implicit reservation is already represented by a SharedResourceUsage, then add a ServiceContextResourceUsage representing this use to it.
- If the implicit reservation is not represented by a SharedResourceUsage, create a SharedResourceUsage and add the existing ResourceUsage and a ServiceContextResourceUsage representing this use to it.

If an existing implicitly reserved NetworkInterface cannot satisfy the need (e.g., because sharing is not allowed, not supported, or not possible), then:

- Reserve a network interface as if the org.davic.net.tuning.NetworkInterfaceController.reserveFor method were called.

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 a NetworkInterface resource for, or is trying to reserve a NetworkInterface 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 implicit reservation persists after the application the resource is reserved for is destroyed, the implementation SHALL continue to use the application identifier of the application for purposes of resource contention resolution until the resource is otherwise released.

- 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 consider the destroyed application to have an effective priority of zero for purposes of resource contention resolution until the resource is otherwise released.

- If the NetworkInterface reservation fails for a ServiceContext.select() method call, the select fails and the appropriate event is generated. If an implicit NetworkInterface reservation is withdrawn prior to being implicitly released, then the service presentation for which it is being used SHALL be terminated and the appropriate event 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 that 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 that 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 manufacturer OCAP applications

NOTE: For avoidance of doubt, this does not mean that the DAVIC resource management mechanisms need to be used by host device manufacturer 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 Multi-Function Hosts 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 a Multi-Function Host 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 Multi-Function Host will determine if the reservation is successful, i.e., the video signal path is configured for cable. If the Multi-Function Host 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

At a minimum, the following resources, where they are visible to both OCAP and non-OCAP environments, SHALL be considered shared:

- Application exclusive key events
- The right to control the resolution of graphics planes
- Tuners that 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 that are not exclusively reserved when an application has focus
**Note:** focus is a shared resource

- The right to control the resolution of graphics planes that are not used by OCAP 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 that are never visible to OCAP applications, (e.g., those used by the implementation for access to MPEG PSI, DSMCC object carousel, AIT, etc.)
- VBI filters that 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 1.0.2] Section.

This section specifies core cable network support functionality including support before a CableCARD device is inserted into an OCAP-compliant set-top box. The baseline functionality is part of the OCAP 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-MHP 1.0.3].

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 Baseline Functionality</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
</tbody>
</table>

20.2 OCAP Specific Requirements

This information extends the specification requirements made to [DVB-MHP 1.0.3].

20.2.1 Boot Process

This subsection details the boot process for OCAP.

The boot process distinguishes between reboot scenarios, which begin at Step 1 in the subsections below, and initialization of the OCAP environment, which begins at Step 3 in Section 20.2.1.1 or Step 4 of Section 20.2.1.2. A CableCARD device may be inserted or removed at any time during the boot process. See Section 20.2.2, item 1 for requirements.

The Host can become disconnected from the network, which can occur inside or outside the consumer premises and can be caused by network failure. In this case, the Host is able to present any abstract services provided by the network when an XAIT was saved in persistent storage when a CableCARD is present, as well as abstract services that are provided by the Host Device Manufacturer. The boot process defined in the following sections SHALL be applied whether or not the Host is connected to Cable. Connection to the cable network is determined by availability of any cable signaling.

Upon initiation of the boot process, the power mode of the set-top terminal defaults to on mode (see Section 25.2.2.7.1), in order to enable video ports and display initialization screens; see [HOST]. Once the initial monitor application launch step in the boot-up process is complete, the set-top terminal will transition to standby mode. Unless explicitly modified (e.g., by monitor application request or by the user pressing the power key), the set-top terminal will remain in standby mode. Changes to the power mode SHALL NOT affect the boot process.

This process is summarized in Figure 20–1.
20.2.1.1 Boot Process - CableCARD device absent

1. **Power Applied or Reboot** - The boot process is started following application of power to the set-top terminal or a software-requested reboot.

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. **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.

5. **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 environment states.

6. **Begin Normal Operation** – see Section 20.2.1.3 below.

20.2.1.2 Boot Process - CableCARD device present

1. **Power Applied or Reboot** - The boot process is started following application of power to the set-top terminal or a software-requested reboot.

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.

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. If the Host is disconnected from the Cable network, then the boot process should proceed to step 7 and attempt to acquire an XAIT that may be stored in persistent memory. If the CableCARD commences a Diffie-Hellman key exchange as defined by [CCCP], the implementation SHALL remove any XAIT previously stored in persistent storage.

6. **Code Download** - If there is a more recent release of OCAP available on the network, upgrade the implementation to it as per Section 20.2.2.7. The set-top terminal SHALL be running the most current OCAP 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. If the implementation detects the OOB channel is inoperable, or after the XAIT maximum cycle time an XAIT has not been received, the XAIT is determined to be unavailable from the network, and the implementation SHALL defer to the XAIT stored in persistent storage, if present; see [HOST] for XAIT storage requirements. If an OOB EAS message is received while waiting for an XAIT, the implementation SHALL process the EAS message as defined by [SCTE 18]. When an XAIT is determined to be unavailable from the network or is not stored in persistent storage, the implementation SHALL skip to step 9. If an XAIT is available, the implementation SHALL update the SI database with signaled abstract services, and it SHALL update the appropriate Applications Database with each application signaled in the XAIT and update stored application information as signaled in the XAIT (refer to Section 11.2.2.3 and Section 12.2.6).

8. **Launch the Initial Monitor Application** - When an XAIT is available but does not contain an Initial Monitor Application entry, the implementation SHALL skip this step. Otherwise, the implementation SHALL launch the Initial Monitor Application with the highest value for unbound_application_descriptor.version_number from the XAIT, unless an application_storage_descriptor is present. In that case, the Initial Monitor Application with the highest launch order signaled by the XAIT SHALL be launched. When an Initial Monitor Application...
is launched, the next step in this boot process SHALL be deferred until one of the following scenarios is completed:

- Five seconds have elapsed since the time the Initial Monitor Applications constructor 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 20.2.2.1.2 and Annex O for invocation details of the `monitorConfiguringSignal` and `monitorConfiguredSignal` methods. If no Initial Monitor Application is signaled, the boot process SHALL continue at Step 10 below.

If the Host is connected to the Cable network and the CableCARD is inserted, then the implementation SHALL present any OOB EAS messages received during this step as per [SCTE 18] and subject to the current power mode.

The Initial Monitor Application MAY launch applications and services before calling the `monitorConfiguredSignal` method.

The Initial Monitor Application MAY query and modify the power mode using the `org.ocap.system.hardware.Host` API (see Annex F).

9. **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 an 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.

10. **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 environment states.

A change in the signaled version of an XAIT could cause an upgrade to some or all of the unbound applications signaled in the XAIT. Changes in the XAIT are managed by the Executive Module; the Monitor Application, if installed, MAY register interest in changes in the Applications Database that result from a revised XAIT by setting an `AppsDatabaseEventListener` (see [DVB-MHP 1.0.3], Annex S), or register interest in the presence of an updated XAIT by setting the AppSignalHandler. If during boot an XAIT was received, but after the boot process and during runtime the XAIT becomes unavailable from signaling for any reason, then the implementation SHALL defer to the XAIT stored in persistent storage as required by [HOST]. XAIT unavailability is determined by absence of XAIT receipt for a time that is greater than the maximum XAIT cycle time defined in Section 11.2.2.3. The implementation SHALL store a signaled XAIT in persistent storage whenever the signaled XAIT version number or CRC fields are different from the same fields in an XAIT previously stored to persistent storage, or when there is no persistently stored XAIT.

11. **Begin Normal Operation** – see Section 20.2.1.3 below.

**NOTE:** 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.

During normal operation, the implementation SHALL monitor the network for a newly signaled or changed XAIT. If an XAIT is detected when previously there was none, the implementation SHALL save the XAIT to persistent storage and re-initialize the OCAP environment per Section 20.2.1.2, Step 4.
20.2.1.3 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. Note that such interaction MAY be limited while the terminal is in standby power mode (see Section 25.2.2.7.1). Normal Operation can be 'CE mode', 'watch TV', or normal cable operation. The implementation MAY present audio/video services at this step as allowed by the security system and without first receiving consumer input. Audio/video services SHALL NOT be presented during the boot process before this step except for EAS purposes as defined by [SCTE 18]. After this, the asynchronous events that could cause the OCAP 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 20.2.1.2 Step 4. If a CableCARD device is removed, the implementation SHALL begin the boot process at Section 20.2.1.1 Step 3.

2. **The Host device or OCAP Application initiates a reboot** - A reboot request SHALL restart the appropriate boot process either at Step 1 or at the "Initialization of OCAP Environment" step, per implementation discretion.

3. **CableCARD reset** - When the CableCARD is reset, the implementation SHALL either treat the reset as a temporary interruption and remain in the Normal Operation state, or reboot. If the implementation supports CableCARD reset without reboot, all implementation-managed CableCARD activities such as XAIT monitoring, SCTE-65 table monitoring, and CAS sessions SHALL be restored by the implementation once the reset is complete.

20.2.1.4 Boot Process Flow Chart (Informative)

The flow chart in Figure 20–1 summarizes the boot process in Section 20.2.1.
An OCAP implementation is required to perform certain operations as part of its baseline functionality and in order to support OCAP applications. A set of conceptual modules is described in this section. These modules are not explicit parts of an OCAP implementation; they are sets of required functionality. The set of conceptual modules described below are:

- Executive Module
- Closed Caption Module
- System Information Module
- CableCARD Interface Resources Module
- Copy Protection Module
- Content Advisory Module
- Download Module
- Watch TV Module

**Figure 20–1 - Boot Process Flow Chart**

**20.2.2 Conceptual Module Descriptions**
- CableCARD Data Channel Module
- Emergency Alert (EAS) Module

20.2.2.1 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

20.2.2.1.1 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 that is launched.

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, 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 for any reason 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 are:

- 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 during the boot process.
- The Initial Monitor Application is destroyed.

20.2.2.1.2 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 implementation 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 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 13.3.8.2 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.
20.2.2.1.3 Management of Stored Applications

The Executive Module is responsible for the management of stored applications signaled in the XAIT as described in Section 12.

20.2.2.2 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 host device SHALL support the closed captioning defined in Section 8.2.4 of [HOST].

20.2.2.3 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 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 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.

20.2.2.4 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].

20.2.2.4.1 Private Host Application on Specific Application Support Resource

[CCIF] supports Private Host Applications. A 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 a Private Host Application and a vendor-specific CableCARD application use them to communicate using APDUs. A Private Host Application has a unique Private Host Application ID so that a vendor-specific CableCARD application can identify it. A Private Host Application reserves a session, associating it with the Private Host Application ID via the sas_connect_rqst() APDU. A Private Host Application can't share a session with another Private Host Application (i.e., one Private Host Application can reserve only one session). A Private Host Application can't close a session, since there is no public APDU to signal a close command from a Host side. A session is closed only when the Host shuts down the power, or when the CableCARD device is removed, or the CableCARD is reset.

An OCAP-J application with MonitorAppPermission("handler.podApplication") permission MAY act as a Private Host Application 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. 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 \texttt{org.ocap.system.SystemModuleRegistrar} described in the following sections. Note that a native resident Private Host Application SHALL be implemented as described in Sections 7.2.1.5.1.4.3 and 10.2.2.7.

20.2.2.4.1.1 Registration

OCAP-J applications MAY act as Private Host Applications by registering with the \texttt{org.ocap.system.SystemModuleRegistrar} via the \texttt{SystemModuleRegistrar.registerSASHandler()} method. This method registers the specified \texttt{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 an \texttt{sas_connect_rqst} APDU to the CableCARD device using the selected session to establish an SAS connection. If an \texttt{sas_connect_cnf} APDU with \texttt{sas_session_status=0x00} returns on the same session from the CableCARD device, the SAS connection is established successfully. See [CCIF] for more information on establishing the SAS connection. If successful, the OCAP implementation SHALL call the \texttt{SystemModuleHandler.ready()} method with a new \texttt{SystemModule} instance. The \texttt{SystemModule} is associated with the same session number as associated with the \texttt{SystemModuleHandler}. If the connection attempt fails, the OCAP implementation SHALL call the \texttt{SystemModuleHandler.ready()} method with a null parameter and unregister the \texttt{SystemModuleHandler} automatically. Only one pair of a \texttt{SystemModule} instance and a \texttt{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 \texttt{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 shuts down the power or the CableCARD device is removed).

If another \texttt{SystemModuleHandler} that has a matching Private Host Application ID has already been registered, it SHALL be unregistered according to Section 20.2.2.4.1.3, and then the specified new \texttt{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 \texttt{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 \texttt{monitorConfiguredSignal()}, Section 20.2.1). 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 \texttt{SystemModuleRegistrar.registerSASHandler()} method in Annex Q.

20.2.2.4.1.2 Communication with CableCARD Device

If registration of a \texttt{SystemModuleHandler} is successful, the OCAP-J application implementing the handler MAY use the \texttt{SystemModule} returned by the \texttt{SystemModuleHandler.ready()} method to send an APDU to the CableCARD device. The \texttt{SystemModule.sendAPDU()} method sends the specified APDU bytes to the CableCARD device according to [CCIF]. The session number associated to the \texttt{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 the 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 SHALL be 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].

For more information, see the description of the SystemModule and the SystemModuleHandler in Annex Q.

20.2.2.4.1.3 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 to 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 MAY 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.

20.2.2.4.2 Man Machine Interface Resource and Application Information Resource


20.2.2.4.2.1 Registration

OCAP-J applications with the MonitorAppPermission("handler.podApplication") permission MAY access these Resources. The MMI Resource always uses the Application Information Resource, so OCAP provides a single API to access these Resources.

An OCAP-J application MAY access 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 application calls 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 MAY 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.

20.2.2.4.2.2 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]. 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 SHALL be 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]. The OCAP implementation SHALL NOT confirm the validity of the specified APDU
parameter, but it SHALL use the apdu_tag value to identify which session should be used.

See also the description of the SystemModule and the SystemModuleHandler in Annex Q.

20.2.2.4.2.3 Unregistration

If the SystemModuleRegistrar.unregisterMMIHandler() method is called, the SystemModuleHandler
instance and the SystemModule instance registered via the registerMMIHandler() method SHALL be
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 deterministically. 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 MAY send and receive APDU for a terminating process until notifyUnregister() method returns. The OCAP implementation SHALL call the unregisterMMIHandler() method with an individual thread to avoid blocking. Note that the session is not closed.

See Annex Q for more details.

20.2.2.5 Copy Protection Module

The Copy Protection Module controls copying of analog or digital content, and the output of content with respect to CCI information.

20.2.2.5.1 General Information

OCAP implementations SHALL fully protect all content as required by the <tru2way> Host Device License Agreement ([HDLA]).

20.2.2.5.1.1 Content Protection Requirements

Copy protection SHALL be implemented in conformance with the OpenCable CableCARD Copy Protection System specification, ([CCCP]). 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]. 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 section 9.1 of [CCCP]. The EMI bits are supplied to any Host digital output ports including the 1394 source port for control of copies made from those outputs.

20.2.2.6 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 APIs for access by applications.

20.2.2.7 Download Module

The download module determines if the OCAP compliant Host Device has a firmware upgrade version that is available via cable download. The download module complies with [CCIF].

20.2.2.8 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. Baseline Watch TV capabilities are channel up and down and numerical channel number entry. As required by the Host [HOST], baseline Watch TV is implemented by the manufacturer. An OCAP application may implement Watch TV functionality by registering for channel up and down and numeric keycodes (VK_CHANNEL_UP, VK_CHANNEL_DOWN, VK_0–9). When an OCAP application has registered interest in these keycodes, the implementation SHALL NOT initiate service selection when these keycodes are received.
20.2.2.9 **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.

20.2.2.10 **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 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 have explicitly reserved these resources prior to the device entering an EAS state, the resources SHALL be taken away using the DAVIC resource negotiation framework. If applications do not voluntarily release resources through the DAVIC resource negotiations, resources SHALL 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 SHALL be terminated due to unavailability of resources. The effect on applications when resources are taken away for EAS handling SHALL 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 disrupts EAS message presentation to the user, the implementation SHOULD make the root containers of all applications invisible. Applications SHALL be notified using the appropriate AWT event notification.

The implementation SHALL NOT allow resources necessary for EAS handling to be acquired by applications while the EAS handling is in progress.

What resources are used for EAS handling is implementation-dependent. The implementation is responsible for making sure that EAS handling conforms to [SCTE 18], 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 MAY be acquired by the implementation. In addition, the implementation MAY 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:

1. Listen for EAS events and check EAS state via the `org.ocap.system.EASManager`;
2. Query and set EAS display attributes using the `org.ocap.system.EASRegistrar`;
3. 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 MAY 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 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.

[SCTE 18] specifies screen representation of an alert text in the alert_text() field in the cable_emergency_alert() message. An OCAP-J application MAY 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.

An OCAP-J application MAY also set an EASHandler. If the alert_priority>11, and the audio is signaled with a private descriptor, the OCAP implementation SHALL call the EASHandler.notifyPrivateDescriptor() method. An OCAP-J application MAY 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 of [SCTE 18]. The OCAP implementation SHALL notify that the specified alert duration finished via the EASHandler.stopAudio() method. If an OCAP-J application doesn't support the private descriptor, the EASHandler.notifyPrivateDescriptor() method SHOULD return false, and the OCAP implementation MAY play detailed channel or proprietary audio.

See Annex Q for more details.
21 MONITOR FUNCTIONALITY

This section contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

This section describes the OCAP 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-MHP 1.0.3] specification.

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>21 Monitor Functionality</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
</tbody>
</table>

21.2 OCAP 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.
- 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 configure presentation of Emergency Alert System messages. (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 Monitor Application MAY configure the logger. (See Section 21.2.1.30.)

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 Application Environment and other applications.

![Figure 21–1 - OCAP Applications](image-url)
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 20.2.2.1.1.

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 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-MHP 1.0.3] Annex S).

Section 20.2.2.1.1 defines how the implementation is responsible for assuring the download and storage of the most recent version of the Monitor Application in the OCAP terminal when it is rebooted. This will assure validity of the Monitor Application, when:

- A brand new OCAP terminal has been installed for the first time
- The OCAP 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 20.2.2.1.1 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 20.2.1.2.

### 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("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.

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 20.2.2.4.1.

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
21.2.1.13 Application Priority Management

A Monitor Application that has `MonitorAppPermission("servicemanager")` permission can set the priority of any other application, except for applications with Monitor Application priority. The `org.ocap.application.AppManagerProxy` interface contains a `setApplicationPriority` method that provides this capability. The priority set by the `setApplicationPriority` method SHALL persist until a new version of the application is signaled, a reboot occurs, or the `setApplicationPriority` 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 kinds of closed-captioning attributes. 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 the `org.ocap.media.ClosedCaptioningControl` class. One of the analog services (CC1 to CC4, T1 to T4) defined in [CEA-608-E] and digital services (Service #1 to #6) defined in [CEA-708-D] 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, [CEA-608-E] 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, [CEA 516] defines another wave format for NABTS teletext and uses line 10 to 20, and [SCTE 20] and [SCTE 21] define 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 [CEA-608-E], Section 9. For Text services (T1 to T4) of [CEA-608-E], one data unit is bytes between a Text service in code and an out code. For generic [CEA-608-E] 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-byte prefix and a 28-byte data block with optional suffix is a VBI data unit. So the VBI data unit for NABTS is fixed 33 bytes length. See also [CEA 516] Section 2. For AMOL, a fixed length bit sequence in one VBI line is a VBI data unit. See [ACN 403-1218-024] 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 an MPEG video, and if a line 21 is specified, the VBI filter retrieves a line 21 data unit from `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):

\[
\text{posFilterDef and posFilterMask == single data unit and posFilterMask}
\]

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):

\[
\text{negFilterDef and negFilterMask != single data unit and negFilterMask}
\]

At a minimum, OCAP SHALL support filtering of CC1, CC2, T1 and T2 data formats in VBI line 21 field 1, and CC3, CC4, T3, T4 and XDS data formats in VBI line 21 field 2, as specified in [CEA-608-E]. 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.

In general, the VBI data filtering API is not intended for synchronized on-screen display of filtered data with video and audio, as filter performance is not guaranteed to be within an acceptable 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-MHP 1.0.3] 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]) that 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.

21.2.1.19.6  CableCARD Reset Event

To receive CableCARD reset events, an application registers a CABLE_CARD_EVENT_LISTENER with the SystemEventManager; see Annex U. There are two notifications given to an application registered to receive CableCARD reset events: CableCARDResetEvent with begin code (CABLECARD_RESET_BEGIN), and CableCARDResetEvent with complete code (CABLECARD_RESET_COMPLETE). When one or more listeners is registered, the implementation SHALL generate a CableCARDResetEvent with begin code when the Host begins a PCMCIA reset process. This can occur when the CableCARD requests a reset by setting the ER bit in the query byte from the CableCARD, or when the Host detects a timeout or other error condition with the CableCARD interface that does not necessitate a Host reset. See [CCIF] for ER bit definition. When the PCMCIA reset completes and the CableCARD sets the CableCARD Ready flag (CR bit in the query byte), the implementation SHALL generate a CableCARDResetEvent with complete code. See [CCIF] for CR bit definition.

Applications that receive a CableCARDResetEvent begin notification should unregister any CableCARD handlers that they registered and cease CableCARD communications. Applications can again register CableCARD handlers when they receive a CableCARDResetEvent complete.

21.2.1.20  System Properties Permissions

A java.util.PropertyPermission with action string "read" shall be granted for applications with MonitorAppPermission("properties") for all properties listed in Table 13–5.

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 13–5.

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">
    <file name="Common.class" size="10"/>
  </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.

### 21.2.1.21.2 Accessing 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:
Figure 21–2 - Registered API Access Process

21.2.1.21.3 Installation

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 14.2.1.16 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 `RegisteredApiManager` 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.

21.2.1.21.4 Access

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 (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")`.

21.2.1.21.5 Persistence

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.

21.2.1.21.6 Removal

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.

21.2.1.21.7 Update

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.

21.2.1.21.8 Security

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 do 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 desire 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

<table>
<thead>
<tr>
<th>Triggers leading to a call to the MediaAccessHandler during the revaluation process</th>
<th>Description</th>
<th>Mandatory/Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMT_CHANGED</td>
<td>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).</td>
<td>Mandatory</td>
</tr>
<tr>
<td>RESOURCE_AVAILABILITY_CHANGED</td>
<td>Access to a resource has changed: lost or free resource.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>NEW_SELECTED_SERVICE</td>
<td>A new service has been selected.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>NEW_SELECTED_SERVICE_COMPONENTS</td>
<td>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). OCAP implementation shall not generate such trigger on a new service selection.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>POWER_STATE_CHANGED</td>
<td>The power state has changed, (e.g., switch to power on).</td>
<td>Optional</td>
</tr>
<tr>
<td>CURRENT_PROGRAM_EVENT_CHANGED</td>
<td>The current program event has changed. The OCAP implementation can generate this trigger in case of SCTE-65 profile 4, 5, and 6.</td>
<td>Optional</td>
</tr>
<tr>
<td>USER_RATING_CHANGED</td>
<td>User preference for rating has been changed.</td>
<td>Optional</td>
</tr>
<tr>
<td>PROGRAM_EVENT_RATING_CHANGED</td>
<td>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.</td>
<td>Optional</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Reason</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BROADCAST_INCONSISTENCY</td>
<td>AlternativeMediaPresentationReason indicating that broadcast information is inconsistent: for example PMT is missing.</td>
</tr>
<tr>
<td>CA_UNKNOWN</td>
<td>AlternativeMediaPresentationReason indicating that media are ciphered and the CA does not correspond to ciphering.</td>
</tr>
<tr>
<td>COMMERCIAL_DIALOG</td>
<td>AlternativeMediaPresentationReason indicating that a user dialog for payment is necessary before media presentation.</td>
</tr>
<tr>
<td>HARDWARE_Resource_NOT_AVAILABLE</td>
<td>AlternativeMediaPresentationReason indicating that hardware resource necessary for presenting service components is not available.</td>
</tr>
<tr>
<td>RATING_PROBLEM</td>
<td>AlternativeMediaPresentationReason indicating that media presentation is not authorized due to invalid rating.</td>
</tr>
<tr>
<td>NO_ENTITLEMENT</td>
<td>AlternativeMediaPresentationReason indicating that service components are ciphered and the user has no entitlement to view them or part of them.</td>
</tr>
</tbody>
</table>

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 preferences specified in Table 21–4.

<table>
<thead>
<tr>
<th>Preference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog Audio</td>
<td>One of &quot;Primary&quot; or &quot;Secondary&quot; indicating the preferred analog audio service. If set to &quot;Secondary&quot;, then the OCAP implementation is responsible for selecting the Secondary Audio Program, if present, for the analog service by default.</td>
</tr>
<tr>
<td>Closed Caption On</td>
<td>One of &quot;On&quot; or &quot;Off&quot;. If set to &quot;Off&quot;, the OCAP implementation SHALL turn off closed captioning by default. Existing presentations SHALL be updated to reflect any change in this preference setting.</td>
</tr>
<tr>
<td>Analog Closed Caption Service</td>
<td>One of &quot;CC1&quot;, &quot;CC2&quot;, &quot;CC3&quot;, &quot;CC4&quot;, &quot;T1&quot;, &quot;T2&quot;, &quot;T3&quot;, or &quot;T4&quot; specifying preferred [CEA-608-E] closed captioning service. Existing presentations SHALL be updated to reflect any change in this preference setting. In the case of an empty or non-matching preference setting, the implementation is responsible for default behavior.</td>
</tr>
<tr>
<td>Digital Closed Caption Service</td>
<td>A decimal number 1-63 indicating a preferred [CEA-708-D] closed captioning service number. Existing presentations SHALL be updated to reflect any change in this preference setting. In the case of an empty or non-matching preference setting, the implementation is responsible for default behavior.</td>
</tr>
</tbody>
</table>
21.2.1.24 Abstract Services

Applications with MonitorAppPermission ("service") can create and manage their own service contexts and the services running in those service contexts. See 10.2.2.3.3, Monitor Application Permissions, and 10.2.2.4, Broadcast Service Selection, for details.

21.2.1.25 Configured

Applications with MonitorAppPermission ("signal.configured") can call org.ocap.OcapSystem.monitorConfiguredSignal() to signal the implementation to resume boot processing after handlers have been set.

21.2.1.26 Storage

Applications with MonitorAppPermission ("storage") are allowed to control persistent storage devices, control access to these devices, and remove content from these devices.

21.2.1.27 Code Download

Applications with MonitorAppPermission ("codeDownload") are allowed to call Host.codeDownload to initiate a download.

21.2.1.28 Power Mode

Applications with MonitorAppPermission ("powerMode") are allowed to call setPowerMode().

21.2.1.29 Cable Environment

Applications with MonitorAppPermission ("environment.selection") can select and deselect the given environment.

21.2.1.30 Logger Configuration

Applications with MonitorAppPermission("logger.config") can configure the logger using the log4j API.
22 OBJECT CAROUSEL


This section describes the technical differences between the [DVB-MHP 1.0.3] object carousel and the OCAP object carousel. When the [DVB-MHP 1.0.3] is read in the context of OCAP, all instances of the MHP terminal SHALL be read as the OCAP terminal. The numbering of the units of this chapter corresponds to the numbering in [DVB-MHP 1.0.3] 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-MHP 1.0.3], 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-MHP 1.0.3] 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 1.0.2] Section.

Section 22 (this section) of OCAP corresponds to Annex B of [DVB-MHP 1.0.3] as follows:

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>MHP Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.1 DVB-GEM and DVB-MHP Specification Correspondence</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>22.2 OCAP Specific Requirements</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>22.2.1 Deviations from the DVB-MHP Specification</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.1 Introduction</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.1.1 Key to notation</td>
<td>Compliance</td>
</tr>
<tr>
<td>22.2.1.1 DSM-CC Sections</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.2 Object Carousel Profile</td>
<td>Extension</td>
</tr>
<tr>
<td>OCAP</td>
<td>[DVB-GEM 1.0.2] Section</td>
<td>GEM Compliance</td>
<td>[DVB-MHP 1.0.3] Section</td>
<td>MHP Compliance</td>
</tr>
<tr>
<td>------</td>
<td>------------------------</td>
<td>----------------</td>
<td>--------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>22.2.1.1 DSM-CC Sections</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.2.1 DSM-CC Sections</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.2.1.1 Sections per TS packet</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.2.2 Data Carousel</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.2.2.1 General</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.2.2.2 DownloadInfoIndication</td>
<td>Compliance</td>
</tr>
<tr>
<td>22.2.1.2 DownloadServerInitiate</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.2.2.3 DownloadServerInitiate</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.2.2.4 ModuleInfo</td>
<td>Compliance</td>
</tr>
<tr>
<td>22.2.1.3 ServiceGatewayInfo</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.2.2.5 ServiceGatewayInfo</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.2.2.6 Download Cancel</td>
<td>Compliance</td>
</tr>
<tr>
<td>22.2.1.4 Object Carousel NSAP</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Extension</td>
<td>B.2.3 The Object Carousel</td>
<td>Extension</td>
</tr>
<tr>
<td>22.2.1.5 CORBA Strings</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>Annex B.2.3.2, CORBA Strings</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.2.4 Stream Events</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.2.5 Assignment and use of transactionId values</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.2.6 Informative Background</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.2.7 DVB semantics of the transactionId field</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.2.8 Mapping of objects to data carousel modules</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.2.9 Compression of modules</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.2.10 Mounting an Object Carousel</td>
<td>Compliance</td>
</tr>
</tbody>
</table>
### 22.2 OCAP Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

#### 22.2.1 Deviations from the DVB-MHP Specification

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

##### 22.2.1.1 DSM-CC Sections

OCAP deviates from [DVB-MHP 1.0.3] 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 1.0.2] Section: Annex B (normative): Broadcast filesystem and trigger transport and contains extensions of [DVB-MHP 1.0.3] Section: B.2.2.3 DownloadServerInitiate.

[DVB-MHP 1.0.3] 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 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 1.0.2] Section: Annex B (normative): Broadcast filesystem and trigger transport and contains extensions of [DVB-MHP 1.0.3] Section: B.2.2.5 ServiceGatewayInfo.

---

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>MHP Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.2.11 Unavailability of a carousel</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.2.12 Delivery of Carousel within multiple services</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.3 AssociationTag Mapping</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.4 Example of an Object Carousel (informative)</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex B (normative): Broadcast filesystem and trigger transport</td>
<td>Compliance</td>
<td>B.5 Caching</td>
<td>Compliance</td>
</tr>
<tr>
<td>22.2.2 Extensions to DVB-MHP</td>
<td>No Corresponding Section</td>
<td>OCAP Specific Extension</td>
<td>No Corresponding Section</td>
<td>No Corresponding Section</td>
</tr>
</tbody>
</table>
In addition to the restrictions in [DVB-MHP 1.0.3], OCAP requires that the first tap in the \texttt{IOP:IOR} SHALL be of type \texttt{BIOP_DELIVERY\_Para\_Use}.

### 22.2.1.4 Object Carousel NSAP

OCAP deviates from [DVB-MHP 1.0.3] 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 \texttt{DownloadServerInitiate} message SHALL NOT include an OCAP NSAP address, see Section 22.2.1.2 for further information.
- The DSM::ServiceLocation field of the \texttt{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

<table>
<thead>
<tr>
<th>Syntax</th>
<th>bits</th>
<th>Type</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCAPcarouselNSAPaddress()</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AFI</td>
<td>8</td>
<td>uimsbf</td>
<td>0x00</td>
<td>NSAP for private use</td>
</tr>
<tr>
<td>Type</td>
<td>8</td>
<td>uimsbf</td>
<td>0x00</td>
<td>Object carousel NSAP Address</td>
</tr>
<tr>
<td>carousel_id</td>
<td>32</td>
<td>uimsbf</td>
<td>+</td>
<td>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.</td>
</tr>
<tr>
<td>specifierType</td>
<td>8</td>
<td>uimsbf</td>
<td>0x01</td>
<td>IEEE OUI</td>
</tr>
<tr>
<td>specifierData {IEEE OUI}</td>
<td>24</td>
<td>uimsbf</td>
<td>0x001000</td>
<td>CableLabs Organization</td>
</tr>
<tr>
<td>ocap_service_location()</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transport_stream_id</td>
<td>16</td>
<td>uimsbf</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>original_network_id</td>
<td>16</td>
<td>uimsbf</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>service_id</td>
<td>16</td>
<td>uimsbf</td>
<td>+</td>
<td>When multiplex type == '01', this field == source_id as per [SCTE 65]. When multiplex type == '10', or multiplex type == '11', this field == program_number as per [ISO 13818-1].</td>
</tr>
<tr>
<td>multiplex_type</td>
<td>2</td>
<td>uimsbf</td>
<td>+</td>
<td>'00'==reserved, '01' == inband, '10'==DSG application tunnel, '11' == OOB</td>
</tr>
<tr>
<td>if (Multiplex_type == '10')</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>14</td>
<td>bslbf</td>
<td>0x3fff</td>
<td></td>
</tr>
<tr>
<td>external_application_id</td>
<td>16</td>
<td>uimsbf</td>
<td>+</td>
<td>Identifies the associated DSG Application ID as per [DSG].</td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>else{</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>30</td>
<td>bslbf</td>
<td>0x3fffffff</td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
22.2.1.5 **CORBA Strings**

In addition to the restrictions in [DVB-MHP 1.0.3], OCAP 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:

```
<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Type</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>context_id</td>
<td>32</td>
<td>uimsbf</td>
<td>0x00010001</td>
<td>IOP::ServiceID CodeSets = 1</td>
</tr>
<tr>
<td>context1_data_length</td>
<td>16</td>
<td>uimsbf</td>
<td>0x0008</td>
<td>length of CodeSetContext</td>
</tr>
<tr>
<td>char_data</td>
<td>32</td>
<td>uimsbf</td>
<td>0x05010001</td>
<td>TCS-C = UTF-8</td>
</tr>
<tr>
<td>wchar_data</td>
<td>32</td>
<td>uimsbf</td>
<td>0x00000000</td>
<td>TCS-W = unknown (i.e., not defined)</td>
</tr>
</tbody>
</table>
```

### 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 1.0.2] section.

The [DVB-MHP 1.0.3] does not specify handling of an out-of-band object carousel. OCAP extends the MHP to define carriage of object carousels out-of-band and received in MPEG flows from the CableCARD. OCAP 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 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], 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]. 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 1.0.2] inclusion of [DVB-MHP 1.0.3], Section 6.2.5.3. For the purposes of OOB object carousel definition, when regarding [DVB-MHP 1.0.3] 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-MHP 1.0.3], 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-MHP 1.0.3] does not specify handling of object carousels over DSG tunnels. OCAP extends DVB-MHP to define carriage of object carousels over DSG application tunnels.

OCAP supports multiple object carousels on multiple DSG application tunnels. OCAP 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 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 expected to contain only MPEG sections with DSG_Carousel_Header structures as defined in Annex E of [DSG]. If the tunnel does not contain MPEG sections with 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 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 a 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).

22.2.2.8 Error Handling for Analog Sources

The tables below are intended to clarify error handling while tuning to analog sources.

<table>
<thead>
<tr>
<th>Method</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>attach(byte[] NSAPAddress)</td>
<td>DSMCCException is thrown if this API is called while tuning to an analog source.</td>
</tr>
<tr>
<td>attach(Locator l)</td>
<td>DSMCCException is thrown if this API is called while tuning to an analog source.</td>
</tr>
<tr>
<td>attach(Locator aDVBService, int aCarouselId)</td>
<td>MPEGDeliveryException is thrown if this API is called while tuning to an analog source.</td>
</tr>
</tbody>
</table>
### Table 22–4 - org.dvb.dsmcc.DSMCCObject Errors While Tuning to an Analog Source

<table>
<thead>
<tr>
<th>Method</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>asynchronousLoad</code> (AsynchronousLoadingEventListener l)</td>
<td>MPEGDeliveryErrorEvent is thrown if tuning from an analog to a digital source.</td>
</tr>
<tr>
<td><code>synchronousLoad()</code></td>
<td>MPEGDeliveryException is thrown if tuning from an analog to a digital source.</td>
</tr>
<tr>
<td><code>loadDirectoryEntry</code> (AsynchronousLoadingEventListener l)</td>
<td>MPEGDeliveryErrorEvent is thrown if tuning from an analog to a digital source.</td>
</tr>
</tbody>
</table>
23 TEXT PRESENTATION

This section contains requirements that are extensions to [DVB-GEM 1.0.2] Section: Annex D (normative): Text presentation and contains extensions of [DVB-MHP 1.0.3] Section: Annex D (normative): Text presentation.

This section addresses the OCAP 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 1.0.2] Section. Section 23 (this section) of OCAP corresponds to Annex D of [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2] as follows:

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>MHP Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>23.1 DVB-GEM and DVB-MHP Specification Correspondence</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>23.2 OCAP Specific Requirements</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>23.2.1 Deviations from the DVB-MHP Specification</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex D (normative): Text presentation</td>
<td>Compliance</td>
<td>D.1 Scope</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex D (normative): Text presentation</td>
<td>Compliance</td>
<td>D.2 Fonts</td>
<td>A subsection is extended</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex D (normative): Text presentation</td>
<td>Compliance</td>
<td>D.2.1 Embedded fonts</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex D (normative): Text presentation</td>
<td>Compliance</td>
<td>D.2.2 Downloaded fonts</td>
<td>A subsection is extended</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex D (normative): Text presentation</td>
<td>Compliance</td>
<td>D.2.2.1 Font technology</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex D (normative): Text presentation</td>
<td>Compliance</td>
<td>D.2.2.2 Font index files</td>
<td>A subsection is extended</td>
</tr>
<tr>
<td>23.2.1.1 Format of File</td>
<td>Annex D (normative): Text presentation</td>
<td>Compliance</td>
<td>D.2.2.2.1 Format of file</td>
<td>Extension</td>
</tr>
<tr>
<td>23.2.1.2 Element Semantics</td>
<td>Annex D (normative): Text presentation</td>
<td>Compliance</td>
<td>D.2.2.2.2 Element semantics</td>
<td>Extension</td>
</tr>
<tr>
<td>23.2.1.3 Example</td>
<td>Annex D (normative): Text presentation</td>
<td>Compliance</td>
<td>D.2.2.2.3 Example</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex D (normative): Text presentation</td>
<td>Compliance</td>
<td>D.2.2.3 Name and location of font index files</td>
<td>A subsection is extended</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>Annex D (normative): Text presentation</td>
<td>Compliance</td>
<td>D.2.2.3.1 General</td>
<td>Compliance</td>
</tr>
<tr>
<td>23.2.1.4 Name of File</td>
<td>Annex D (normative): Text presentation</td>
<td>Compliance</td>
<td>D.2.2.3.2 Name of file</td>
<td>Extension</td>
</tr>
</tbody>
</table>
### 23.2 OCAP Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

#### 23.2.1 Deviations from the DVB-MHP Specification

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

##### 23.2.1.1 Format of File

This subsection is compliant with [DVB-GEM 1.0.2] Section: Annex D (normative): Text presentation and contains extensions of [DVB-MHP 1.0.3] Section: D.2.2.2.1 Format of file.

This section extends Section D.2.2.2.1 of the [DVB-GEM 1.0.2].

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 1.0.2] Section: Annex D (normative): Text presentation and complies with [DVB-MHP 1.1.3] Section: D.2.2.2.2 Element semantics.

This section extends Section D.2.2.2.2 of the [DVB-GEM 1.0.2].

**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 1.0.2] Section: Annex D (normative): Text presentation and contains extensions of [DVB-MHP 1.0.3] Section: D.2.2.2.3 Example.

This section extends Section D.2.2.2.3 of the [DVB-GEM 1.0.2].
The OCAP document type declaration SHALL be:

```xml
<!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 1.0.2] Section: Annex D (normative): Text presentation.

This section extends Section D.2.2.3.2 of the [DVB-GEM 1.0.2].

The name of the OCAP font index file SHALL be:

`ocap.fontindex`
24 EXTENSIONS

This section contains requirements that are extensions to [DVB-GEM 1.0.2] 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-MHP 1.0.3]. However they are outside the scope of the [DVB-MHP 1.0.3] 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 1.0.2] Section. Section 24 (this section) of OCAP corresponds to Annex H of [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2] as follows:

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 Extensions</td>
<td>Annex H (normative): Extensions</td>
<td>Extension</td>
</tr>
<tr>
<td>24.1 DVB-GEM and DVB-MHP</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
<tr>
<td>Specification Correspondence</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24.2 OCAP Specific Requirements</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
</tbody>
</table>

24.2 OCAP Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

24.2.1 Extensions to DVB-GEM (Normative)

Private protocols and APIs MAY extend the technology required by OCAP.

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 1.0.2] Section: Annex G (normative): Minimum platform capabilities and are extensions of [DVB-MHP 1.0.3] Section: Annex G (normative): Minimum Platform Capabilities.

The minimum platform capabilities are those features that SHALL be supported by an OCAP 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 1.0.2] Section. Section 25 (this section) of OCAP corresponds to Annex G of [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2] as follows:

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>MHP Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>25.1 DVB-GEM and DVB-MHP Specification Correspondence</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>25.2 OCAP Specific Requirements</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>25.2.1 Deviations from the DVB-MHP Specification</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>G.1 Graphics</td>
<td>A subsection is extended</td>
<td>G.1 Graphics</td>
<td>A subsection is extended</td>
</tr>
<tr>
<td>25.2.1.1 Device Capabilities</td>
<td>G.1.1 Device resolution for Standard Definition</td>
<td>Extension</td>
<td>G.1.1 Device capabilities</td>
<td>Extension</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>G.1.3 Minimum Colour Lookup Table</td>
<td>Compliance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>G.1.2 Device Resolution for Standard Definition</td>
<td>Compliance</td>
<td>G.1.2 Video presentation capabilities</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>No Corresponding Section</td>
<td>G.1.3 Image processing capabilities</td>
<td>Compliance</td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>No Corresponding Section</td>
<td>G.1.4 Alpha capabilities</td>
<td>Compliance</td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>No Corresponding Section</td>
<td>G.1.5 Colour capabilities</td>
<td>Compliance</td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>No Corresponding Section</td>
<td>G.1.6 MPEG I frame and Video drips</td>
<td>Compliance</td>
<td></td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>G.2 Audio</td>
<td>Compliance</td>
<td>G.2 Audio</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>G.3 Video</td>
<td>Compliance</td>
<td>G.3 Video</td>
<td>Compliance</td>
</tr>
<tr>
<td>No Corresponding Section</td>
<td>G.4 Resident fonts and text rendering</td>
<td>Compliance</td>
<td>G.4 Resident fonts and text rendering</td>
<td>Compliance</td>
</tr>
</tbody>
</table>
25.2 OCAP Specific Requirements

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

25.2.1 Deviations from the DVB-MHP Specification

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

25.2.1.1 Device Capabilities

This section extends Annex G.1.1 of the [DVB-MHP 1.0.3].

**NOTE:** Annex G.1.1 of [DVB-GEM 1.0.2] has a bug that references the wrong bullet point item in MHP Annex G.1.1.

References to the **MHP Terminal** SHALL be replaced with references to the **OCAP 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 - Sets 1 and 2**

<table>
<thead>
<tr>
<th></th>
<th>Set #1</th>
<th>Set #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>HGraphicsDevice</td>
<td>640x480</td>
<td>960x540</td>
</tr>
<tr>
<td>HVideDevice</td>
<td>720x480 (according to ITU-R BT.601)</td>
<td>720x480 (according to ITU-R BT.601)</td>
</tr>
<tr>
<td>HBackgroundDevice</td>
<td>Either 640x480 or 720x480</td>
<td>Either 640x480 or 720x480</td>
</tr>
</tbody>
</table>

- 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 - Sets 3 and 4**

<table>
<thead>
<tr>
<th></th>
<th>Set #3</th>
<th>Set #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>HGraphicsDevice</td>
<td>640x480</td>
<td>960x540</td>
</tr>
<tr>
<td>HVideDevice</td>
<td>1920x1080 (according to ITU-R BT.709)</td>
<td>1920x1080 (according to ITU-R BT.709)</td>
</tr>
<tr>
<td>HBackgroundDevice</td>
<td>No I-frame supported</td>
<td>No I-frame supported</td>
</tr>
</tbody>
</table>

- 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 HVideDevice changes or configuration of the HVideDevice 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 HVideDevice 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 - Sets 5 through 8**

<table>
<thead>
<tr>
<th></th>
<th>Set #5</th>
<th>Set #6</th>
<th>Set #7</th>
<th>Set #8</th>
</tr>
</thead>
<tbody>
<tr>
<td>HGraphicsDevice</td>
<td>640x480</td>
<td>640x480</td>
<td>960x540</td>
<td>960x540</td>
</tr>
<tr>
<td>HVideDevice</td>
<td>No video input</td>
<td>No video input</td>
<td>No video input</td>
<td>No video input</td>
</tr>
<tr>
<td>HBackgroundDevice</td>
<td>Either 640x480 or 720x480</td>
<td>1920x1080</td>
<td>Either 640x480 or 720x480</td>
<td>1920x1080</td>
</tr>
</tbody>
</table>
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
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 \texttt{HBackgroundDevice} and the \texttt{HBackgroundDevice} is configured to be full screen. An I-frame on an \texttt{HBackgroundDevice} SHALL be able to be simultaneously displayed with video on an \texttt{HVideoDevice}, while decoding of an I-frame while simultaneously decoding video is an implementation option.

\textbf{NOTE}: An application may get the current resolution of \texttt{HBackgroundDevice} via \texttt{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 1.0.2] Section: G.5 Input events and are extensions of [DVB-MHP 1.0.3] Section: G.5 Input events.

This section extends Annex G.5 of [DVB-GEM 1.0.2]:

- OCAP extends the full set of HAVi remote key events. These extensions are defined in Annex D.2.2.1, OCAP Extensions to HReEvent. The minimal set of key events codes that SHALL be supported for OCAP is defined in Table 25–5 (i.e., Table 25–5 replaces Table G.3 of [DVB-MHP 1.0.3]).
- 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 sent to OCAP applications when the corresponding key is pressed by a user.

#### Table 25–5 - Minimum Keycode Set

<table>
<thead>
<tr>
<th>Key</th>
<th>KeyEvent</th>
<th>Mandatory Ordinary Keycodes</th>
<th>System Keycodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power On/Off</td>
<td>VK_POWER</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Channel Down</td>
<td>VK_CHANNEL_DOWN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Channel Up</td>
<td>VK_CHANNEL_UP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0-9</td>
<td>VK_0 to VK_9</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Up</td>
<td>VK_UP</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Down</td>
<td>VK_DOWN</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Left</td>
<td>VK_LEFT</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Right</td>
<td>VK_RIGHT</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Select</td>
<td>VK_ENTER</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Volume Down</td>
<td>VK_VOLUME_DOWN</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Volume Up</td>
<td>VK_VOLUME_UP</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Mute Volume</td>
<td>VK_MUTE</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Key</th>
<th>KeyEvent</th>
<th>Mandatory Ordinary Keycodes</th>
<th>System Keycodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pause</td>
<td>VK_PAUSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Play</td>
<td>VK_PLAY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop</td>
<td>VK_STOP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Record</td>
<td>VK_RECORD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast Forward</td>
<td>VK_FAST_FWD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rewind</td>
<td>VK_REWIND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Electronic Program Guide</td>
<td>VK_GUIDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RF Bypass</td>
<td>VK_RF_BYPASS</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Menu</td>
<td>VK_MENU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td>VK_INFO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exit</td>
<td>VK_EXIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last</td>
<td>VK_LAST</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function Key 0</td>
<td>VK_COLORED_KEY_0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function Key 1</td>
<td>VK_COLORED_KEY_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function Key 2</td>
<td>VK_COLORED_KEY_2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function Key 3</td>
<td>VK_COLORED_KEY_3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page Up</td>
<td>VK_PAGE_UP</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Page Down</td>
<td>VK_PAGE_DOWN</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Next Favorite Channel</td>
<td>VK_NEXT_FAVORITE_CHANNEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>On Demand</td>
<td>VK_ON_DEMAND</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 25.2.2 Extensions to DVB-GEM (Normative)

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

The following requirements are normative for OCAP. They are specific to OCAP and are not contained within the [DVB-MHP 1.0.3].

#### 25.2.2.1 Multi-Window System

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: G.7 Other resources and are extensions of [DVB-MHP 1.0.3] 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 13.3.6.2. The implementation SHALL withdraw focus from an application when any of the following events occur:

---

<table>
<thead>
<tr>
<th>Key</th>
<th>KeyEvent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home</td>
<td>VK_HOME</td>
</tr>
<tr>
<td>Left</td>
<td>VK_LEFT</td>
</tr>
<tr>
<td>Up</td>
<td>VK_UP</td>
</tr>
<tr>
<td>RIGHT</td>
<td>VK_RIGHT</td>
</tr>
<tr>
<td>DOWN</td>
<td>VK_DOWN</td>
</tr>
<tr>
<td>.</td>
<td>VK_COMMA</td>
</tr>
<tr>
<td>:</td>
<td>VK_PERIOD</td>
</tr>
<tr>
<td>/</td>
<td>VK_SLASH</td>
</tr>
<tr>
<td>0 - 9</td>
<td>VK_0 - VK_9</td>
</tr>
<tr>
<td>;</td>
<td>VK_SEMICOLON</td>
</tr>
<tr>
<td>=</td>
<td>VK_EQUALS</td>
</tr>
<tr>
<td>A - Z</td>
<td>VK_A - VK_Z</td>
</tr>
<tr>
<td>[</td>
<td>VK_OPEN_BRACKET</td>
</tr>
<tr>
<td>\</td>
<td>VK_BACK_SLASH</td>
</tr>
<tr>
<td>]</td>
<td>VK_CLOSE_BRACKET</td>
</tr>
<tr>
<td>0 - 9</td>
<td>VK_NUMPAD0 - VK_NUMPAD9</td>
</tr>
<tr>
<td>F1</td>
<td>VK_F1</td>
</tr>
<tr>
<td>F2</td>
<td>VK_F2</td>
</tr>
<tr>
<td>F3</td>
<td>VK_F3</td>
</tr>
<tr>
<td>F4</td>
<td>VK_F4</td>
</tr>
<tr>
<td>DELETE</td>
<td>VK_DELETE</td>
</tr>
<tr>
<td>HELP</td>
<td>VK_HELP</td>
</tr>
<tr>
<td>QUOTE</td>
<td>VK_QUOTE</td>
</tr>
</tbody>
</table>
• 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 instances. An activable HScene is a visible and active (as described in HScene.setActive javadoc) HScene which has explicitly requested focus at least one time.
• The focused HScene is an activable HScene that is assigned input focus. There is only one focused HScene at a time.
• When an HScene becomes activable, it is added to the end of the activable HScene list.
• When an activable HScene is assigned focus, it is moved to the beginning of the activable HScene list.
• When an HScene is no longer activable, it is removed from the activable HScene list.
• When the focused HScene is no longer activable, the implementation SHALL withdraw focus from the focused HScene and assign 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 assign 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 1.0.2] Section.

The resources as specified for all OCAP profiles in [HOST] are REQUIRED.

25.2.2.3 Persistent Storage

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

The OCAP implementation SHALL provide a mechanism for logically separate storage of:

• the platform's image as defined in [CCIF], 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 20.2.2.7), persistent storage APIs, (see [DVB-MHP 1.0.3], Section 11.5.6), and boot-up processing. (See Section 20.2.1, step #6.)

There SHALL be a minimum of 32 MBs of persistent storage for application code and data (Flash, disk, etc.) available. See Section 12.
25.2.2.4 Threading

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: G.7 Other resources and are extensions of [DVB-MHP 1.0.3] Section: G.7 Other resources.

The row specifying requirements for "Application accessible DVB-J threads" of [DVB-MHP 1.0.3] Table G.5 SHALL be considered replaced by Table 25–7.

Table 25–7 - Minimum requirements for other resources for conformance purposes

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
</table>
| Application accessible (OCAP-J threads) | At least \( T_{\text{shared}} \) threads shared among all concurrent non-manufacturer applications. At least \( T_{\text{service}} \) threads shared among all applications signaled as part of the same broadcast service. At least \( T_{\text{app}} \) threads per application, so long as the shared minimums are not exceeded. Threads created by the platform and used to call methods of the application, threads created by host-manufacturer applications, and completed threads are excluded from these numbers. There shall be no limits on the number of threads that may be created and destroyed over the total execution time of the Host. | \( \begin{array}{|c|c|}\hline
T_{\text{shared}} & 128 \\
T_{\text{service}} & 32 \\
T_{\text{app}} & 32 \\
\hline
\end{array} \) |

25.2.2.5 RAM

This subsection contains requirements that are extensions to [DVB-GEM 1.0.2] Section: G.6.

OCAP implementations SHALL provide at least 64MB of memory to support OCAP-J applications. At least this amount of memory SHALL be available across all applications, regardless of number.

The following minimum memory requirements are defined for OCAP terminals. All are to be measured in the \texttt{startXlet()} method of one or more OCAP-J test applications. The implementation SHALL:

Successfully load any arbitrary 1MB (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 classes comprising the OCAP-J test application(s) are included within this 1MB of Java class files, and the test application(s) will cause additional classes to be loaded sufficient to meet the 1MB requirement if necessary.

The implementation SHALL supply enough memory to do the above and individually each of the following. These individual tests are not required to run concurrently, and it is expected that the memory for each test is recovered before the next test is run.

1. Successfully create up to 10 Java byte arrays of arbitrary length with a combined total of up to 60,817,408 (58M) entries.
2. Successfully create 46 instances of \texttt{org.dvb.ui.DVBBufferedImage} of type \texttt{DVBBufferedImage.TYPE\_ADVANCED}, and 640x480 pixels in size.
3. Successfully load from one or more files into memory 3456 seconds of audio encoded at 128 kbit/s (where kbit/s is as used in [ETSI TR 101 154]). It SHALL be measured using files that do not include any optional extension fields.

4. Successfully allocate up to 10 arrays of \texttt{java.lang.Object} of arbitrary length with a combined total of up to 3,538,944 entries, and fill each array element with a distinct instance of \texttt{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.

\subsection*{25.2.2.6 Keyboard Input Events}

The keyboard input events as identified in the \texttt{java.awt.event.KeyEvent} class, as specified in [JSR 217], SHALL be supported.

\subsection*{25.2.2.7 Power Control}

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

\subsubsection*{25.2.2.7.1 Power Input}

OpenCable requires support for query of the AC power supplied to the OCAP 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 \texttt{standby} mode or is \texttt{on}. When the terminal is \texttt{on}, its full facilities are available. When the terminal is in \texttt{standby} mode, all audio and video outputs SHALL be inactive. OCAP applications SHALL NOT be terminated while the terminal is in \texttt{standby} mode, for purposes such as EPG update, software maintenance, and quick refresh, when normal power mode is re-established. When components such as front panel, remote control, or keyboard remain operational in \texttt{standby} mode, input from those components SHALL be sent to applications running during \texttt{standby} mode in the same manner as when the terminal is \texttt{on}.

\textbf{NOTE:} (informative) Applications that receive key input in \texttt{standby} mode are recommended to be aware of power mode and modify behavior accordingly, in an application specific fashion. Applications MAY lose focus due to resource ownership changes caused by a transition to \texttt{standby} mode.

\subsection*{25.2.3 Extensions to DVB-GEM (Informative)}

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

The following information is informative to OCAP. It attempts to clarify issues in the [DVB-MHP 1.0.3], as well as provide additional information to the normative requirements above.

The OCAP 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].

\subsubsection*{25.2.3.1 General Hardware Configuration}

This subsection contains OCAP-specific requirements that do not correspond to any [DVB-GEM 1.0.2] Section.

OpenCable recommends the following hardware configuration with respect to processor and memory capabilities.
25.2.3.1.1 Processor Capability

An OCAP-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, and also in excess of processor usage in the provision of core functionality described in [HOST]. This capability includes support for video demux, decode and presentation.

25.2.3.1.2 Minimum Memory Configuration

An OCAP-compliant terminal will need to provide some amount of memory in excess of any configuration required for OCAP, including video decoding and graphics display.

25.2.3.1.2.1 ROM

Flash Memory SHOULD be partitioned in 128 KByte sectors. Optional protection of Boot Sectors is acceptable to reduce implementation cost.

25.2.3.1.2.2 NVRAM

OpenCable recommends that the OCAP 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 1.0.2] Section.

OpenCable recommends support for the following peripheral devices.

25.2.3.2.1 Keyboard Device Capabilities

OpenCable recommends support for keyboard input.

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.
Annex A  Multi-Function Implementation Details

A.1  Overview

This annex of the OCAP Specification provides implementation details for multi-function devices incorporating environments other than OCAP. OCAP enables manufacturers to incorporate OCAP into a Multi-Function Host, i.e., a device in which cable services as provided by OCAP are just one set of services. Other services might be DVD playback, presentation of terrestrial TV, or other manufacturer capability.

A.2  Operational Rules

This section defines the operational rules for implementers of a Multi-Function Host.

A.2.1  Cross-environment Applications

By default, host device manufacturer applications are not capable of operating as Cross-environment Applications when Cable Environment is the Selected Environment (see Section 10.2.2.4). Applications that can run as Cross-Environment applications outside of their home environment are defined as such in their home environment. See also Section 10.2.2.7.2.

A.2.1.1  Restrictions on non-OCAP utility applications

Non-OCAP cross-environment applications are limited to providing a User Interface to perform the following functions:

a. Configuration of Video Settings, including but not limited to: Sharpness, Color, Picture Size (Wide, Zoom, 4:3), Digital Noise Reduction, Tint, Brightness, Contrast, Auto Picture Mode.

b. Configuration of Audio Settings, including but not limited to: Equalizer, Surround Sound, MTS, Auto Volume, Internal Mute, and Auto Mode.

c. Configuration of Initial Setup, including but not limited to: Time Set, Channel Scan, V-Chip, Blue Screen, and Captioning.

d. Configuration of Timers, including but not limited to: Sleep Timer, On Timer, and Off Timer.

e. Configuration of Input, including but not limited to: Select Signal Source, Edit name of source.

A.2.1.2  Restrictions on cross-service applications

Cross-service Applications may operate only if:

a. A Viewer affirmatively makes an explicit direct request to use the cross-service Application for each instance in which the cross-service Application runs. A generalized opt-in by the end-user or Viewer is not a direct request.

b. The cross-service Application SHALL time out and terminate within 30 seconds of the Viewer's last affirmative input entered to use the cross-service Application.

c. The cross-service Application may draw onscreen graphics overtop of the Cable Environment, provided that applications in the Cable Environment are correctly redrawn after the cross-service Application is no
longer visible.

d. Any visual element of the cross-service Application is displayed only in a single rectangular area of the display screen and does not frame the Cable Service; for instance, by displaying an 'L-shaped' User Interface composed of vertical and horizontal rectangles;

e. The cross-service Application does not scale the presentation of Cable Services; for instance, by decreasing the display region occupied by the Cable Service.

f. The cross-service Application does not affect any operation or presentation of the Cable Service, except as described in 10.2.2.4.1.

g. The cross-service Application remains subject to the rules and policies of its home Environment, and does not obtain additional access to cable resources. For example, a cross-service Application will not have access to the cable return channel.

A.2.2 Multi-Function Host Handling of Services

This section defines how Cable Service is treated by the Multi-Function Host.

Every Multi-Function Host must pass through and display all Cable Services in their entirety without interruption, impairment, or modification, in accordance with the terms of Section 11.2 of [HDLA].

A.2.2.1 Behavioral Restrictions

The Multi-Function Host SHALL NOT:

a. Insert any information or materials, including, but not limited to, advertising, interactive content (such as Advanced Television Enhancement Forum triggers and related content), or overlays into or over Cable Service, regardless of whether or not any such insertion is perceptible to the Cable Operator or the Viewer.

b. Remove or alter any information or materials, including, but not limited to, copy protection instructions, ratings information, interactive content, overlays, advertising, content or other network or Cable Operator-inserted materials, from Cable Service, regardless of whether or not any such removal or alteration is perceptible to the Cable Operator or the Viewer.

c. Synchronize or permit the synchronization of Licensee or third party advertising or content with Cable Services without the Cable Operator's express written consent.

A.2.3 Access to OCAP applications

A Viewer SHALL be able to access OCAP applications from any Selected Environment in which the Multi-Function Host permits any other navigation or menu application to operate. Each remote control or other user interface for a Multi-Function Host SHALL make OCAP applications at least as accessible and useable by the Viewer as any other navigation or menu application. An example of an OCAP application is an Electronic Program Guide installed into the Host by the network operator on whose network the Multi-Function Host is connected. This standard SHALL include, but is not limited to, the following requirements:

a. Multi-Function Hosts SHALL enable Viewers to access OCAP applications with as few steps (i.e., clicks of a button, spoken commands, screen menus) as is minimally necessary to access any other navigation or menu application. For purposes of measuring the accessibility of other navigation or menu application, a Viewer's means of accessing a navigation or menu application other than an OCAP application SHALL include any options that may be programmed by the Viewer, whether or not such options are advertised by Licensee.
b. The end-user and Viewer SHALL be able to access OCAP applications in any environment in which another navigation or menu application can be accessed.

c. If any button or physical interface is labeled "guide" or a similar term or symbol, either that button or interface SHALL direct the Viewer to an OCAP application, or the button and physical interfaces used to access the OCAP application SHALL be labeled "Cable Guide."

d. The button(s) and other physical interface(s) used by the Viewer to select a OCAP application SHALL be at least the same size and have at least as prominent a location and appearance as the button(s) and other physical interface(s) used to access other navigation or menu applications.

e. Multi-Function Hosts SHALL execute all key events in a nondiscriminatory manner. Regardless of which type of keycode interface is chosen by Licensee, the labels of the key functionality as presented to the Viewer and end-user SHALL conform to the labeling requirements contained in the OCAP Specification.

f. The Multi-Function Host SHALL provide access to OCAP applications through an auxiliary method set forth in the OCAP Specification, whether or not other methods are also provided.

g. The Multi-Function Host SHALL communicate all OCAP keycodes to OCAP when cable is the Selected Environment. When cable is not the Selected Environment, those keycodes that are reserved by OCAP applications will be communicated to the cable environment.
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

Annex B (this section) of OCAP corresponds to Annex R of [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2] as follows:

**Table B–1 - Correlation between OCAP and [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3]**

<table>
<thead>
<tr>
<th>OCAP Section</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>Compliance/Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annex B OCAP Specific Network APIs</td>
<td>Annex R, DVB-J Return Channel Connection Management API</td>
<td></td>
<td>Extension to GEM</td>
</tr>
<tr>
<td>Annex B.1 DVB-GEM and DVB-MHP Specification Correspondence</td>
<td>No corresponding section</td>
<td></td>
<td>No corresponding section</td>
</tr>
<tr>
<td>Annex B.2 OCAP Specific Requirements</td>
<td>No corresponding section</td>
<td></td>
<td>No corresponding section</td>
</tr>
<tr>
<td>Annex B.2.1 Deviations to DVB-MHP</td>
<td>No corresponding section</td>
<td></td>
<td>Extension to GEM</td>
</tr>
<tr>
<td>Annex B.2.1.1 Extensions to the DVB-J Return Channel Management API</td>
<td>getType() method</td>
<td></td>
<td>Extension to GEM</td>
</tr>
<tr>
<td>Annex B.2.2 Extensions to DVB-MHP (Normative)</td>
<td>No corresponding section</td>
<td></td>
<td>No corresponding section</td>
</tr>
<tr>
<td>Annex B.2.3 Java.net Usage</td>
<td>No corresponding section</td>
<td></td>
<td>Extension to GEM</td>
</tr>
</tbody>
</table>

This section corresponds to Annex R of the [DVB-MHP 1.0.3]. The OpenCable Application Platform is in complete compliance with the API, org.dvb.net.rc, specified in this section.

B.2 OCAP 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. In particular, DVB addresses CATV, PSTN/ISDN, DECT, GSM, LMDS, and SMATV networks. OCAP 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] 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].

B.2.2 **Extensions to DVB-MHP (Normative)**

The following information is normative for OCAP. It is specific to OCAP and is not contained within the [DVB-MHP 1.0.3].

B.2.3 **Java.net Usage**

OCAP 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]. An application may use java.net to access this communication channel.

B.2.3.1 **InetAddress.getLocalHost Clarification**

The behavior of `java.net.InetAddress.getLocalHost()` as defined in [JSR 217] is extended and clarified as follows:

- Subject to granted privileges, this method SHALL return the `InetAddress` corresponding to the OCAP return channel ([SCTE 55-1]/[SCTE 55-2] or DOCSIS) network interface.
- The `InetAddress` returned by this method SHALL always represent the current IP address assigned to the network interface.

**NOTE:** (Informative) The `java.net.NetworkInterface` class can be used to obtain any additional optional network interfaces and their corresponding `InetAddress`es.
Annex C  Permissions

This annex of the OCAP Specification specifies APIs for Control Access permissions and Monitor Application permission.

In addition to the DVB APIs, this annex of OCAP specifies APIs for privileged service information and selection permissions.

C.1  DVB-GEM and DVB-MHP Specification Correspondence

Annex C (this annex) of OCAP corresponds to Annex T of [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2] as follows:

Table C–1 - Correlation between OCAP and [DVB-GEM 1.0.2]

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annex C Permissions</td>
<td>Annex T (normative):</td>
<td>Compliance</td>
</tr>
<tr>
<td></td>
<td>Permissions</td>
<td></td>
</tr>
<tr>
<td>Annex C.1 DVB-GEM and DVB-MHP Specification Correspondence</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
<tr>
<td>Annex C.2 OCAP Specific Requirements</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
<tr>
<td>Annex C.2.1 Extensions to DVB-MHP (Normative)</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
<tr>
<td>Annex C.2.1.1 Monitor Application Permission</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
</tbody>
</table>

C.2  OCAP Specific Requirements

C.2.1  Extensions to DVB-MHP (Normative)

This information extends the specification requirements made to the [DVB-MHP 1.0.3].

The following information is normative for OCAP. It is specific to OCAP and is not contained within the [DVB-MHP 1.0.3].

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 complies with the HAVi specification referenced by [DVB-MHP 1.0.3] as follows:

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>MHP Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annex D HAVi Level 2 User Interface</td>
<td>VOID</td>
<td>Unknown</td>
<td>Annex V</td>
<td>Unknown</td>
</tr>
<tr>
<td>Annex D.1 DVB-GEM and DVB-MHP Specification Correspondence</td>
<td>No Corresponding Section</td>
<td>Unknown</td>
<td>No Corresponding Section</td>
<td>Unknown</td>
</tr>
<tr>
<td>Annex D.2 OCAP Specific Requirements</td>
<td>No Corresponding Section</td>
<td>Unknown</td>
<td>No Corresponding Section</td>
<td>Unknown</td>
</tr>
<tr>
<td>Annex D.2.1 Deviations from the HAVi Specification</td>
<td>No Corresponding Section</td>
<td>Unknown</td>
<td>No Corresponding Section</td>
<td>Unknown</td>
</tr>
<tr>
<td>Annex D.2.1.1 org.havi.ui.event.HRcCapabilities</td>
<td>No Corresponding Section</td>
<td>Unknown</td>
<td>No Corresponding Section</td>
<td>Unknown</td>
</tr>
<tr>
<td>Annex D.2.1.2 org.havi.ui.event.HEventRepresentation</td>
<td>No Corresponding Section</td>
<td>Unknown</td>
<td>No Corresponding Section</td>
<td>Unknown</td>
</tr>
<tr>
<td>Annex D.2.2 Extensions to DVB-MHP (Normative)</td>
<td>No Corresponding Section</td>
<td>Unknown</td>
<td>No Corresponding Section</td>
<td>Unknown</td>
</tr>
<tr>
<td>Annex D.2.2.1 OCAP Extensions to HRcEvent</td>
<td>No Corresponding Section</td>
<td>Unknown</td>
<td>No Corresponding Section</td>
<td>Unknown</td>
</tr>
</tbody>
</table>

The OpenCable Application Platform is in complete compliance with the HAVi API, org.havi.ui, as specified in this section.

D.2  OCAP 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-MHP 1.0.3].

The following information is normative for OCAP. It is specific to OCAP and is not contained within the [DVB-MHP 1.0.3].

D.2.2.1 **OCAP Extensions to H RcEvent**

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.)

```java
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 UI and UI Event APIs

This section presents the `org.ocap.ui` and `org.ocap.ui.event` APIs.

| Table E–1 - Correlation between OCAP and [DVB-GEM 1.0.2] |
|---------------------------------|-------------------------------|
| OCAP                           | [DVB-GEM 1.0.2] Section       | GEM Compliance             |
| Annex E OCAP UI and UI Event API | No Corresponding Section     | OCAP-Specific Extension    |

**Package org.ocap.ui**

Extensions to HAVi User Interface functionality.

See:

Description

**Interface Summary**

<table>
<thead>
<tr>
<th>HSceneBinding</th>
<th>Defines a binding between the area on a display that an HScene is (or will) occupy and the associated application.</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSceneChangeRequestHandler</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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. |

**Package org.ocap.ui Description**

Extensions to HAVi User Interface functionality.

Since:

OCAP 1.0
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

<table>
<thead>
<tr>
<th>OcapAppAttributes</th>
<th>getAppAttributes()</th>
</tr>
</thead>
</table>
|                   | Gets the attributes of the application associated with the HScene.

<table>
<thead>
<tr>
<th>HScreenRectangle</th>
<th>getRectangle()</th>
</tr>
</thead>
</table>
|                   | Gets the rectangle in normalised co-ordinates that the HScene is using or has requested to use.

Method Detail

getRectangle

HScreenRectangle getRectangle()

Gets the rectangle in normalised 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

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>testMove</td>
<td>boolean testMove(HSceneBinding move, HSceneBinding[] currentScenes)</td>
<td>Tests whether an HScene move request can be allowed or not.</td>
</tr>
<tr>
<td>testOrder</td>
<td>boolean testOrder(HSceneBinding[] currentScenes, int currentOrder, int newOrder)</td>
<td>Tests if an HScene z-order change request can be made or not.</td>
</tr>
<tr>
<td>testShow</td>
<td>boolean testShow(HSceneBinding newScene, HSceneBinding[] oldScenes)</td>
<td>Tests whether an HScene display request can be allowed or not.</td>
</tr>
</tbody>
</table>

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,
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 `currentScenes` 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.
org.ocap.ui
Class HSceneManager

defined by java.lang.Object

  org.ocap.ui.HSceneManager
public abstract class HSceneManager
    extends java.lang.Object

This class represents a manager that lets an application register a handler to requested HScene changes within a logical HScreen composited with all HScenes. In addition, HScene z-ordering can be queried using this manager.

Constructor Summary

<table>
<thead>
<tr>
<th>protected</th>
<th>HSceneManager()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Protected default constructor.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>int</th>
<th>getAppHSceneLocation()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the current HScene z-order location for a specific HScene.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static OcapAppAttributes[]</th>
<th>getHSceneOrder()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the current HScene z-ordering.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static HSceneManager</th>
<th>getInstance()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the singleton instance of the HScene manager.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setHSceneChangeRequestHandler(HSceneChangeRequestHandler handler)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lets an application add itself as the HScene change request handler.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

HSceneManager
protected HSceneManager()
  Protected default constructor.

Method Detail

getInstance

public static HSceneManager getInstance()
  Gets the singleton instance of the HScene manager. The singleton MAY be implemented using application or implementation scope.
Returns:
The HScene manager.

`setHSceneChangeRequestHandler`

```
public void setHSceneChangeRequestHandler(HSceneChangeRequestHandler handler)
```

Lets an application add itself as the HScene change request handler. If a handler is already registered when
this method is called, it is replaced with the parameter handler.

**Parameters:**

- `handler` - HSceneChangeRequestHandler for requests to HScene z-ordering changes. If this parameter is
  null the current handler is removed.

**Throws:**

- `java.lang.SecurityException` - if the caller does not have
  MonitorAppPermission("handler.resource").

`getHSceneOrder`

```
public static OcapAppAttributes[] getHSceneOrder()
```

Gets the current HScene z-ordering. The array of attributes returned is ordered increasing in z-order where
the first entry (0) corresponds to an HScene on top and the last entry is on bottom.

**Returns:**

Array of application attributes corresponding to HScene instances in z-order.

**Throws:**

- `java.lang.SecurityException` - if the caller does not have
  MonitorAppPermission("handler.resource").

`getAppHSceneLocation`

```
public int getAppHSceneLocation()
```

Gets the current HScene z-order location for a specific HScene. Applications can call this to determine
where their HScene is located.

**Returns:**

HScene z-order location for the calling application. The value is ordered increasing in z-order where 0 is on
top and all other values are in increasing order below the top. A value of -1 indicates the HScene has not
been ordered.
**Package org.ocap.ui.event**

Extensions to HAVi User Interface Event classes, including OCAP specific remote control events and multiscreen management events.

**See:**

**Description**

### Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCRcEvent</td>
<td>The OCAP remote control event class.</td>
</tr>
</tbody>
</table>

**Package org.ocap.ui.event Description**

Extensions to HAVi User Interface Event classes, including OCAP specific remote control events and multiscreen management events.

**Since:**

OCAP 1.0
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 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.HEventRepresentation.

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.

See Also:
  Serialized Form

Field Summary

| static int | OCRC_FIRST | Marks the first integer id for the range of OCAP remote control key codes. |
| static int | OCRC_LAST  | Marks the last integer id for the range of OCAP remote control key codes. |
| static int | VK_APPS   | The 'apps' key code. |
| static int | VK_BACK   | The 'back' key code. |
| static int | VK_CC     | The 'Closed Caption' key code. |
| static int | VK_EXIT   | The 'exit' key code. |
| static int | VK_FORWARD| The 'forward' key code. |
## Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int VK_INSTANT_REPLAY</td>
<td>The 'instant replay' key code.</td>
</tr>
<tr>
<td>static int VK_LAST</td>
<td>The 'last' key code.</td>
</tr>
<tr>
<td>static int VK_LINK</td>
<td>The 'link' key code.</td>
</tr>
<tr>
<td>static int VK_LIST</td>
<td>The 'list' key code.</td>
</tr>
<tr>
<td>static int VK_LIVE</td>
<td>The 'live' key code.</td>
</tr>
<tr>
<td>static int VK_LOCK</td>
<td>The 'lock' key code.</td>
</tr>
<tr>
<td>static int VK_MENU</td>
<td>The 'menu' key code.</td>
</tr>
<tr>
<td>static int VK_NEXT_DAY</td>
<td>The guide 'next day' key code.</td>
</tr>
<tr>
<td>static int VK_NEXT_FAVORITE_CHANNEL</td>
<td>The 'next favorite channel' key code.</td>
</tr>
<tr>
<td>static int VK_ON_DEMAND</td>
<td>The 'on demand' key code.</td>
</tr>
<tr>
<td>static int VK_PINP_DOWN</td>
<td>The 'picture-in-picture down' key code.</td>
</tr>
<tr>
<td>static int VK_PINP_MOVE</td>
<td>The 'picture-in-picture move' key code.</td>
</tr>
<tr>
<td>static int VK_PINP_UP</td>
<td>The 'picture-in-picture up' key code.</td>
</tr>
<tr>
<td>static int VK_PREV_DAY</td>
<td>The guide 'previous day' key code.</td>
</tr>
<tr>
<td>static int VK_RC_LOW_BATTERY</td>
<td>The ‘RC low battery’ key code Indicates that the remote control battery is low.</td>
</tr>
<tr>
<td>static int VK_RESERVE_1</td>
<td>The 'reserved' key code number 1.</td>
</tr>
<tr>
<td>static int VK_RESERVE_2</td>
<td>The 'reserved' key code number 2.</td>
</tr>
<tr>
<td>static int VK_RESERVE_3</td>
<td>The 'reserved' key code number 3.</td>
</tr>
<tr>
<td>static int VK_RESERVE_4</td>
<td>The 'reserved' key code number 4.</td>
</tr>
<tr>
<td>static int VK_RESERVE_5</td>
<td>The 'reserved' key code number 5.</td>
</tr>
<tr>
<td>static int VK_RESERVE_6</td>
<td>The 'reserved' key code number 6.</td>
</tr>
</tbody>
</table>
Field Summary

static int VK_RF_BYPASS
   The 'RF Bypass' key code.

static int VK_SETTINGS
   The 'settings' key code.

static int VK_SKIP
   The 'skip' key code.

static int VK_USER
   The 'user' key code.

static int VK_ZOOM
   The 'zoom' key code.

Fields inherited from class org.havi.ui.event.HRcKeyEvent

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_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_ALT, VK_B, VK_BACK_QUOTE, VK_BACK_SLASH, VK_BACK_SPACE, VK_C,
VK_CANCEL, VK_CAPS_LOCK, VK_CLEAR, VK_CLOSE_BRACKET, VK_COMMAND, VK_CONTROL,
VK_CONVERT, VK_D, VK_DECIMAL, VK_DELETE, VK_DOWN, VK_E, VK_END,
VK_EJECT, VK_EQUALS, VK_ESCAPE, VK_F, VK_F1, VK_F10, VK_F11, VK_F12,
VK_F2, VK_F3, VK_F4, VK_F5, VK_F6, VK_F7, VK_F8, VK_F9, VK_FINAL, VK_G, VK_H,
VK_HELP, VK_HOME, VK_I, VK_INSERT, VK_J, VK_K, VK_KANA, VK_KANJI, VK_L,
VK_LEFT, VK_M, VK_META, VK_MODECHANGE, VK_MULTIPLY, VK_N, VK_NONCONVERT,
VK_NUM_LOCK, VK_NUMPAD0, VK_NUMPAD1, VK_NUMPAD2, VK_NUMPAD3, VK_NUMPAD4,
VK_NUMPAD5, VK_NUMPAD6, VK_NUMPAD7, VK_NUMPAD8, VK_NUMPAD9, VK_Q,
VK_OPEN_BRACKET, VK_P, VK_PAGE_DOWN, VK_PAGE_UP, VK_PAUSE, VK_PERIOD,
VK_PRINTSCREEN, VK_Q, VKQUOTE, VK_R, VK_RIGHT, VK_S, VK_SCROLL_LOCK,
VK_SEMICOLON, VK_SEPARATOR, VK_SHIFT, VK_SLASH, VK_SPACE, VK_SUBTRACT, VK_T,
VK_TAB, VK_U, VK_UNDEFINED, VK_UP, VK_V, VK_W, VK_X, VK_Y, VK_Z

Fields inherited from class java.awt.event.InputEvent

ALT_MASK, BUTTON1_MASK, BUTTON2_MASK, BUTTON3_MASK, CTRL_MASK, META_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, id, ITEM_EVENT_MASK, KEY_EVENT_MASK, MOUSE_EVENT_MASK, MOUSE_MOTION_EVENT_MASK, RESERVED_ID_MAX, TEXT_EVENT_MASK, WINDOW_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, getKeyModifiersText, getKeyText, isActionKey, paramString, setKeyChar, setKeyCode, setModifiers

### Methods inherited from class java.awt.event.InputEvent

consume, getModifiers, getWhen, isAltDown, isConsumed, isControlDown, isMetaDown, isShiftDown

### Methods inherited from class java.awt.event.ComponentEvent

gGetComponent

### Methods inherited from class java.awt.AWTEvent

getID, toString

### Methods inherited from class java.util.EventObject

getSource
Field Detail

OCRC_FIRST
public static final int OCRC_FIRST
    Marks the first integer id for the range of OCAP remote control key codes.
    See Also: Constant Field Values

VK_RF_BYPASS
public static final int VK_RF_BYPASS
    The 'RF Bypass' key code. Indicates a user request to bypass the set-top by passing the RF input directly to
    the set-top RF output (toggle).
    See Also: Constant Field Values

VK_EXIT
public static final int VK_EXIT
    The 'exit' key code. Indicates a user request to exit the current application.
    See Also: Constant Field Values

VK_MENU
public static final int VK_MENU
    The 'menu' key code. Indicates a user request for an on-screen menu (toggle).
    See Also: Constant Field Values

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, Constant Field Values

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, Constant Field Values
VK_APPS

public static final int VK_APPS
    The 'apps' key code. Indicates a user request for applications.
    See Also:
    Constant Field Values

VK_LINK

public static final int VK_LINK
    The 'link' key code. Indicates a user request for launching linked content.
    See Also:
    Constant Field Values

VK_LAST

public static final int VK_LAST
    The 'last' key code. Indicates a user request for tuning to the last channel tuned.
    See Also:
    Constant Field Values

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, Constant Field Values

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, Constant Field Values

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.
    See Also:
    Constant Field Values

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.).
    See Also:
    Constant Field Values

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.
See Also:  
Constant Field Values

**VK_RESERVE_1**

```java
public static final int VK_RESERVE_1
```

The 'reserved' key code number 1. Reserved for future use.

See Also:  
Constant Field Values

**VK_RESERVE_2**

```java
public static final int VK_RESERVE_2
```

The 'reserved' key code number 2. Reserved for future use.

See Also:  
Constant Field Values

**VK_RESERVE_3**

```java
public static final int VK_RESERVE_3
```

The 'reserved' key code number 3. Reserved for future use.

See Also:  
Constant Field Values

**VK_RESERVE_4**

```java
public static final int VK_RESERVE_4
```

The 'reserved' key code number 4. Reserved for future use.

See Also:  
Constant Field Values

**VK_RESERVE_5**

```java
public static final int VK_RESERVE_5
```

The 'reserved' key code number 5. Reserved for future use.

See Also:  
Constant Field Values

**VK_RESERVE_6**

```java
public static final int VK_RESERVE_6
```

The 'reserved' key code number 6. Reserved for future use.

See Also:  
Constant Field Values

**VK_LOCK**

```java
public static final int VK_LOCK
```

The 'lock' key code. Indicates a user request to lock the current program.

See Also:  
Constant Field Values

**VK_SKIP**

```java
public static final int VK_SKIP
```
The 'skip' key code. Indicates a user request to skip the current program.

See Also:
Constant Field Values

**VK_LIST**

```java
public static final int VK_LIST
```

The 'list' key code. Indicates a user request to list the current program.

See Also:
Constant Field Values

**VK_LIVE**

```java
public static final int VK_LIVE
```

The 'live' key code. Indicates a user request to view live programs.

See Also:
Constant Field Values

**VK_ON_DEMAND**

```java
public static final int VK_ON_DEMAND
```

The 'on demand' key code. Indicates a user request to access on demand functions.

See Also:
Constant Field Values

**VK_PINP_MOVE**

```java
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.

See Also:
Constant Field Values

**VK_PINP_UP**

```java
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.

See Also:
Constant Field Values

**VK_PINP_DOWN**

```java
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.

See Also:
Constant Field Values

**VK_INSTANT_REPLAY**

```java
public static final int VK_INSTANT_REPLAY
```

The 'instant replay' key code. Indicates a user request to invoke the instant replay feature.

See Also:
Constant Field Values
VK_RC_LOW_BATTERY
public static final int VK_RC_LOW_BATTERY
    The 'RC low battery' key code Indicates that the remote control battery is low. Generated automatically by the remote, there are no corresponding buttons on the remote. Informative Note: Dialog to the user should indicate this is for the STB remote.
    See Also:
    Constant Field Values

VK_USER
public static final int VK_USER
    The 'user' key code. Indicates a user request to switch to the next user profile if multiple user profiles exist.
    See Also:
    Constant Field Values

VK_CC
public static final int VK_CC
    The 'Closed Caption' key code. Indicates a user request to toggle closed caption on/off.
    See Also:
    Constant Field Values

OCRC_LAST
public static final int OCRC_LAST
    Marks the last integer id for the range of OCAP remote control key codes.
    See Also:
    Constant Field Values

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 Hardware API

Table F–1 - Correlation between OCAP and [DVB-GEM 1.0.2]

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annex F, OCAP Hardware API</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
</tbody>
</table>

Package org.ocap.hardware

The org.ocap.hardware package provides representations of several aspects of the hardware.

See:

Description

Interface Summary

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE1394Node</td>
<td>This interface represents the information on a 1394 node.</td>
</tr>
<tr>
<td>PowerModeChangeListener</td>
<td>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).</td>
</tr>
</tbody>
</table>

Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CopyControl</td>
<td>This class represents the copy control information on the analog and digital A/V outputs of the OCAP terminal.</td>
</tr>
<tr>
<td>Host</td>
<td>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.</td>
</tr>
<tr>
<td>VideoOutputPort</td>
<td>An object of this class represents an analog or digital video output of the OCAP terminal.</td>
</tr>
</tbody>
</table>

Package org.ocap.hardware Description

The org.ocap.hardware package provides representations of several aspects of the hardware.
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

<table>
<thead>
<tr>
<th>protected</th>
<th>CopyControl()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not use.</td>
<td></td>
</tr>
</tbody>
</table>

Method Summary

static 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.

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 to which the returned CCI value applies. 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.
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

<table>
<thead>
<tr>
<th>static int</th>
<th>FULL_POWER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power mode constant for normal &quot;on&quot; mode.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>LOW_POWER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power mode constant for &quot;standby&quot; mode.</td>
<td></td>
</tr>
</tbody>
</table>

Constructor Summary

protected Host()
A constructor of this class.

Method Summary

  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).

  void codeDownload()
  This method initiates a download of the operating software in the Host as
  specified by [CCIF2.0].

  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.

  java.lang.String getID()
  Get a human-readable string representing the ID of this Host.

  static Host getInstance()
  This method returns a singleton system-wide instance of the Host class.

  int getPowerMode()
Method Summary

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.lang.String</td>
<td>getReverseChannelMAC()</td>
<td>Gets the MAC address used by the Host for reverse channel unicast communications.</td>
</tr>
<tr>
<td>boolean</td>
<td>getRFBypass()</td>
<td>Queries whether RF Bypass is currently enabled.</td>
</tr>
<tr>
<td>boolean</td>
<td>getRFBypassCapability()</td>
<td>Returns capability of RF bypass control on the host.</td>
</tr>
<tr>
<td>java.util.Enumeration</td>
<td>getVideoOutputPorts()</td>
<td>This method returns a java.util.Enumeration of VideoOutputPort instances representing all video output ports physically present on the device.</td>
</tr>
<tr>
<td>boolean</td>
<td>isACOutletPresent()</td>
<td>Query whether there is an AC Outlet on the STB.</td>
</tr>
<tr>
<td>void</td>
<td>reboot()</td>
<td>This method initiates a reboot of the Host device.</td>
</tr>
<tr>
<td>void</td>
<td>removePowerModeChangeListener(PowerModeChangeListener l)</td>
<td>Removes the previously-added PowerModeChangeListener.</td>
</tr>
<tr>
<td>void</td>
<td>removeXAIT()</td>
<td>Removes the XAIT saved to persistent storage.</td>
</tr>
<tr>
<td>void</td>
<td>setACOutlet(boolean enable)</td>
<td>Switch power to AC Outlet, if present, On (true) or Off (false) NOTE: AC Outlet refers to an external power plug on the STB.</td>
</tr>
<tr>
<td>void</td>
<td>setPowerMode(int mode)</td>
<td>Transition the power mode of the system to the given mode.</td>
</tr>
<tr>
<td>void</td>
<td>setRFBypass(boolean enable)</td>
<td>Enables or disables RF Bypass.</td>
</tr>
</tbody>
</table>

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.

See Also:
Constant Field Values

LOW_POWER
public static final int LOW_POWER
Power mode constant for "standby" mode.

See Also:
Constant Field Values
### Constructor Detail

**Host**

```java
protected Host() {
    // A constructor of this class. An application must use the getInstance() method to create an instance.
}
```

### Method Detail

**getInstance**

```java
public static Host getInstance() {
    // This method returns a singleton system-wide instance of the Host class.
    Returns:
    a singleton Host instance.
}
```

**getID**

```java
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**

```java
public int getPowerMode() {
    // Returns:
    the current power mode of the box (for example LOW_POWER).
    See Also:
    FULL_POWER, LOW_POWER
}
```

**getReverseChannelMAC**

```java
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. The format of the String returned
    // SHALL be six pairs of characters where each pair represents a hexadecimal byte value of the address and
    // where each pair is separated by a colon. For example "0D:0E:0F:10:11:12". The first byte representation in
    // the String starting at location 0 SHALL be the most significant byte in the address.
    Returns:
    MAC address of the Host.
}
```

**addPowerModeChangeListener**

```java
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

```java
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

```java
public java.util.Enumeration getVideoOutputPorts()
```

This method returns a java.util.Enumeration of VideoOutputPort instances representing all video output ports physically present on the device. The returned Enumeration SHALL reflect a 1 to 1 mapping between VideoOutputPort instances and physical video output ports. For example, 2 HDMI output ports driven by the same controller would report two distinct VideoOutputPort instances of type AV_OUTPUT_PORT_TYPE_HDMI. This method SHALL report all VideoOutputPort instances regardless of the enabled or disabled status of the port.

**Returns:**

the java.util.Enumeration of VideoOutputPort instances.

reboot

```java
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 MAY clean up resources in the SystemEventListener.notifyEvent method call. After the SystemEventListener.notifyEvent method call returns, the Host device SHALL continue the reboot following the boot process described in the Boot Process Section of this specification.

**Throws:**

java.lang.SecurityException - if the caller does not have the MonitorAppPermission("reboot").

codeDownload

```java
public void codeDownload()
```

This method initiates a download of the operating software in the Host as specified by [CCIF2.0].

**Throws:**

java.lang.SecurityException - if the caller does not have MonitorAppPermission("codeDownload").

isACOutletPresent

```java
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

```java
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.IllegalArgumentException - if this method is called when there is no AC Outlet.

```java
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.IllegalArgumentException - if this method is called when there is no AC Outlet.
```

```java
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.
```

```java
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 standby 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.
```

```java
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 standby mode, thereby totally bypassing the host.

**Parameters:**
enable - If true, RF Bypass will be enabled. Otherwise it will be disabled.

**Throws:**
java.lang.IllegalArgumentException - if the host doesn’t support RF bypass.
```

```java
removeXAIT
public void removeXAIT()
Removes the XAIT saved to persistent storage. If no XAIT is present in persistent storage this method does nothing successfully. This method SHALL NOT affect a cached XAIT and any running applications.

**Throws:**
java.lang.SecurityException - if the calling application is not granted MonitorAppPermission("storage").
```

```java
setPowerMode
public void setPowerMode(int mode)
```
Transition the power mode of the system to the given mode.
If the power mode is already in the target mode, this method SHALL do nothing. Setting host power mode
to low-power SHALL NOT disrupt any ongoing recording. In devices where a separate power mode is
maintained for standby recordings, setting the power mode to low-power SHALL transition to standby-
recording power mode when a recording is in progress.

A change of power mode SHALL be communicated to installed PowerModeChangeListener.

Parameters:
mode - The new power mode for the system.

Throws:
java.lang.IllegalArgumentException - if mode is not one of FULL_POWER or LOW_POWER
java.lang.SecurityException - if the caller does not have
MonitorAppPermission("powerMode")
org.ocap.hardware

Interface IEEE1394Node

public interface IEEE1394Node

This interface represents the information on a 1394 node.

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte[] getEUI64()</td>
<td></td>
<td>Returns the value of EUI-64 of the 1394 node.</td>
</tr>
<tr>
<td>java.lang.String getModelName()</td>
<td></td>
<td>Returns the value of MODEL NAME TEXTUAL DESCRIPTOR of the 1394 node.</td>
</tr>
<tr>
<td>short[] getSubunitType()</td>
<td></td>
<td>Returns the list of subunitTypes supported by the 1394 node.</td>
</tr>
<tr>
<td>java.lang.String getVendorName()</td>
<td></td>
<td>Returns the value of VENDOR NAME TEXTUAL DESCRIPTOR of the 1394 node.</td>
</tr>
</tbody>
</table>

**Method Detail**

getEUI64

```java
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
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").
}
```

gVendorName

```java
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

```java
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

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>powerModeChanged</code></td>
</tr>
</tbody>
</table>

```java
void powerModeChanged(int newPowerMode)
```

Called when the power mode changes (for example from full to low power).

### Method Detail

**powerModeChanged**

```java
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.utilEnumeration 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 |
|           | AV Output Port Type 1394 (Firewire) |
| static int | AV_OUTPUT_PORT_TYPE_BB |
|           | AV Output Port Type Baseband (RCA connector) |
| static int | AV_OUTPUT_PORT_TYPE_COMPONENT_VIDEO |
|           | AV Output Port Type Component Video |
| static int | AV_OUTPUT_PORT_TYPE_DVI |
|           | AV Output Port Type DVI (Panel Link, HDCP) |
| static int | AV_OUTPUT_PORT_TYPE_HDMI |
|           | AV Output Port Type HDMI |
| static int | AV_OUTPUT_PORT_TYPE_INTERNAL |
|           | AV Output Port Type Internal (integrated/internal display) |
| static int | AV_OUTPUT_PORT_TYPE_RF |
|           | AV Output Port Type RF channel 3/4 |
| static int | AV_OUTPUT_PORT_TYPE_SVIDEO |
|           | AV Output Port Type S-Video |
| static int | CAPABILITY_TYPE_DTCP |
|           | AV Output Port Capability Type DTCP |
| static int | CAPABILITY_TYPE_HDCP |
|           | AV Output Port Capability Type HDCP |
Field Summary

<table>
<thead>
<tr>
<th>static int</th>
<th>CAPABILITY_TYPE_RESOLUTION_RESTRICTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AV Output Port Capability Type Resolution Restriction for HD Video</td>
</tr>
</tbody>
</table>

Constructor Summary

<table>
<thead>
<tr>
<th>protected VideoOutputPort()</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCAP applications SHALL NOT use this constructor - it is provided for internal use by the OCAP implementation.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>abstract void disable()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable the video output port, that is, prevent the video output from this port.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>abstract void enable()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable the video output port, that is, allow the video output from this port.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>abstract IEEE1394Node[] getIEEE1394Node()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the list of IEEE1394Node corresponding to all the 1394 nodes that were discovered by the OpenCable Host device.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>abstract java.awt.Dimension getResolution()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query the number of horizontal pixels and vertical lines for the Output Type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>abstract int getType()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the type of this VideoOutputPort.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>abstract java.lang.Object queryCapability(int capabilityType)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query the value related to specified capabilityType.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>abstract void selectIEEE1394Sink(byte[] eui64, short subunitType)</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>abstract boolean status()</th>
</tr>
</thead>
<tbody>
<tr>
<td>This method returns a current status of this video output port.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>public static final int AV_OUTPUT_PORT_TYPE_RF</th>
</tr>
</thead>
<tbody>
<tr>
<td>AV Output Port Type RF channel 3/4</td>
</tr>
</tbody>
</table>

See Also:

Constant Field Values
AV_OUTPUT_PORT_TYPE_BB
public static final int AV_OUTPUT_PORT_TYPE_BB
    AV Output Port Type Baseband (RCA connector)
    See Also:
    Constant Field Values

AV_OUTPUT_PORT_TYPE_SVIDEO
public static final int AV_OUTPUT_PORT_TYPE_SVIDEO
    AV Output Port Type S-Video
    See Also:
    Constant Field Values

AV_OUTPUT_PORT_TYPE_1394
public static final int AV_OUTPUT_PORT_TYPE_1394
    AV Output Port Type 1394 (Firewire)
    See Also:
    Constant Field Values

AV_OUTPUT_PORT_TYPE_DVI
public static final int AV_OUTPUT_PORT_TYPE_DVI
    AV Output Port Type DVI (Panel Link, HDCP)
    See Also:
    Constant Field Values

AV_OUTPUT_PORT_TYPE_COMPONENT_VIDEO
public static final int AV_OUTPUT_PORT_TYPE_COMPONENT_VIDEO
    AV Output Port Type Component Video
    See Also:
    Constant Field Values

AV_OUTPUT_PORT_TYPE_HDMI
public static final int AV_OUTPUT_PORT_TYPE_HDMI
    AV Output Port Type HDMI
    See Also:
    Constant Field Values

AV_OUTPUT_PORT_TYPE_INTERNAL
public static final int AV_OUTPUT_PORT_TYPE_INTERNAL
    AV Output Port Type Internal (integrated/internal display)
    See Also:
    Constant Field Values

CAPABILITY_TYPE_DTCP
public static final int CAPABILITY_TYPE_DTCP
    AV Output Port Capability Type DTCP
    See Also:
    Constant Field Values
CAPABILITY_TYPE_HDCP

public static final int CAPABILITY_TYPE_HDCP

    AV Output Port Capability Type HDCP

    See Also:
    Constant Field Values

CAPABILITY_TYPE_RESOLUTION_RESTRICTION

public static final int CAPABILITY_TYPE_RESOLUTION_RESTRICTION

    AV Output Port Capability Type Resolution Restriction for HD Video

    See Also:
    Constant Field Values

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

capability

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.IllegalArgumentException - 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 Host Device License Agreement [HDLA]. Note that FCC may provide rules for port disabling.

    Throws:
    java.lang.SecurityException - if the caller has not been granted
    MonitorAppPermission("setVideoPort")
    java.lang.IllegalArgumentException - 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).
• CAPABILITY_TYPE_HDCP-java.lang.Boolean which indicates HDCP is available (TRUE).
• CAPABILITY_TYPE_RESOLUTION_RESTRICTION-java.lang.Integer which indicates the restricted pixel resolution for HD video on the analog component video port Returns (-1) for any VideoOutputPort type other than AV_OUTPUT_PORT_TYPE_COMPONENT_VIDEO. For VideoOutputPort type AV_OUTPUT_PORT_TYPE_COMPONENT_VIDEO, returns the integer product of the horizontal resolution and the vertical resolution (h x v) used in the display of a "Constrained Image" which shall be no more than 520,000.

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.IllegalArgumentException - if eui64 is not valid.

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.IllegalArgumentException - 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.

getResolution

public abstract java.awt.Dimension getResolution()

Query the number of horizontal pixels and vertical lines for the Output Type.

Returns:
The Dimension object representing the number of horizontal pixels (width) and vertical lines (height) the Output Type is currently displaying.
Annex G  OCAP Application API

This section presents the `org.ocap.application` APIs.

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annex G, OCAP Application API</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
</tbody>
</table>

Package org.ocap.application

This package contains APIs for controlling the lifecycle of applications.

See:

Description

Interface Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppFilterHandler</td>
<td>Application programs can provide an implementation of this interface to an AppFilter to make part of decision for AppFilter.accept.</td>
</tr>
<tr>
<td>AppSignalHandler</td>
<td>This interface represents a handler that can be registered in order to receive a notification whenever the XAIT version changes.</td>
</tr>
<tr>
<td>OcapAppAttributes</td>
<td>This interface represents various information about an application registered in the AppsDatabase.</td>
</tr>
<tr>
<td>SecurityPolicyHandler</td>
<td>This interface provides a callback handler to modify the Permissions granted to an application to be launched.</td>
</tr>
</tbody>
</table>

Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppFilter</td>
<td><code>AppFilter</code> provides a means of filtering AppIDs.</td>
</tr>
<tr>
<td>AppManagerProxy</td>
<td>This class represents the application manager functionality used by the Monitor Application.</td>
</tr>
<tr>
<td>AppPattern</td>
<td>AppPattern is an element that constitutes an AppFilter.</td>
</tr>
<tr>
<td>OcapIxcPermission</td>
<td>This class represents access to the inter-xlet communication registry.</td>
</tr>
<tr>
<td>PermissionInformation</td>
<td>This class contains information to allow the monitor application to choose the permissions to grant to an application.</td>
</tr>
</tbody>
</table>

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 Signaling of Unbound Applications). 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 XAIT stated in this Specification.

**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.

**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.

```java
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;
        }
    }
}
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.

```java
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());
        }
```

```java
        return true;
    }
}
```
/* 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 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 AppPatterns. 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:

```java
import org.ocap.application.*;
import org.dvb.application.AppID;

AppManagerProxy am = ...;
AppPattern[] patterns = {
    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

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AppFilter</strong></td>
<td>Constructs an empty AppFilter.</td>
</tr>
<tr>
<td><strong>AppFilter(AppPattern[] patterns)</strong></td>
<td>Constructs an AppFilter with initial AppPatterns.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean <strong>accept(AppID appID)</strong></td>
<td>Returns whether this AppFilter accepts the given AppID.</td>
</tr>
<tr>
<td>void <strong>add(AppPattern pattern)</strong></td>
<td>Adds an AppPattern to this AppFilter.</td>
</tr>
<tr>
<td>java.util.Enumeration <strong>getAppPatterns()</strong></td>
<td>Returns the AppPatterns in this AppFilter.</td>
</tr>
<tr>
<td>boolean <strong>remove(AppPattern pattern)</strong></td>
<td>Removes an AppPattern that equals to pattern in this AppFilter.</td>
</tr>
<tr>
<td>void <strong>setAskHandler(AppFilterHandler handler)</strong></td>
<td>Sets the handler to call when accept hits an AppPatterns with action AppPattern.ASK.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Constructor Detail

**AppFilter**

```java
public AppFilter()
    Constructs an empty AppFilter.
```

**AppFilter(AppPattern[] patterns)**

```java
public AppFilter(AppPattern[] patterns)
    Constructs an AppFilter with initial AppPatterns.
```

Parameters:
patterns - AppPatterns to constitute an AppFilter.
Method Detail

**getAppPatterns**

```java
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**

```java
public boolean accept(AppID appID)
```

Returns whether this AppFilter accepts the given AppID.

**Specified by:**
accept in class AppsDatabaseFilter

**Parameters:**
appID - an AppID to test.

**Returns:**
true if appID passes this filter.

**add**

```java
public void add(AppPattern pattern)
```

Adds an AppPattern to this AppFilter.

**Parameters:**
pattern - the AppPattern to add

**remove**

```java
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**

```java
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>accept</td>
<td>boolean accept(AppID appID, AppPattern matchingItem)</td>
</tr>
</tbody>
</table>

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(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 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 the section 10 Application Model and 11 Application Signaling in this specification for details.

Constructor Summary

<table>
<thead>
<tr>
<th>protected</th>
<th>AppManagerProxy()</th>
</tr>
</thead>
</table>
|           | This is a constructor of this class.

Method Summary

<table>
<thead>
<tr>
<th>java.util.Properties</th>
<th>getAddressingProperties()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the addressing properties previously registered by the registerAddressingProperties method.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static AppManagerProxy</th>
<th>getInstance()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This method returns the sole instance of the AppManagerProxy class.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>java.util.Properties</th>
<th>getSecurityAddressableAttributes()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the security system Host addressable attributes queried by the implementation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int[]</th>
<th>getSupportedApplicationTypes()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This method returns the set of application types supported by this OCAP implementation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>registerAddressingProperties(java.util.Properties properties, boolean persist, java.util.Date expirationDate)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Registers addressing properties used for comparison when an addressing_descriptor from an AIT or XAIT is evaluated.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>registerUnboundApp(java.io.InputStream xait)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This method registers new unbound application entries.</td>
</tr>
</tbody>
</table>
Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void removeAddressingProperties(String[] properties)</td>
<td>Removes addressing properties set by the registerAddressingProperties method.</td>
</tr>
<tr>
<td>void setAppFilter(AppsDatabaseFilter filter)</td>
<td>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.</td>
</tr>
<tr>
<td>void setApplicationPriority(int priority, AppID appId)</td>
<td>This method sets the priority for the application.</td>
</tr>
<tr>
<td>void setAppSignalHandler(AppSignalHandler handler)</td>
<td>This method sets an instance of a class that implements the AppSignalHandler interface.</td>
</tr>
<tr>
<td>void setSecurityPolicyHandler(SecurityPolicyHandler handler)</td>
<td>This method sets an instance of a class that implements the SecurityPolicyHandler interface.</td>
</tr>
<tr>
<td>void unregisterUnboundApp(int serviceId, AppID appid)</td>
<td>This method unregisters an unbound application from the AppsDatabase.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, 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(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("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 1.0.2 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")

```java
unregisterUnboundApp

public void unregisterUnboundApp(int serviceId,
                                  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")

```java
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**

```java
public void setApplicationPriority(int priority, 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 changed 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 currently set at monitor application priority.

**registerAddressingProperties**

```java
public void registerAddressingProperties(java.util.Properties properties, boolean persist, java.util.Date expirationDate)
```

Registers addressing properties used for comparison when an addressing_descriptor from an AIT or XAIT is evaluated. The implementation SHALL maintain a set of properties registered by any application. The implementation SHALL adhere to the following rules in order when registering each property passed in the properties parameter:
- When a property contains a value that is not an instance of java.lang.String the property is ignored.
- When a property key is 0 length the property is ignored.
- When a property key is a duplicate of a Java system property the property is ignored.
- When a property key is a duplicate of a property previously registered by this method it is ignored.
- When a property key is a duplicate of an addressable attribute retrieved from the security system the property is ignored.
- When a property key is not registered and the value is not a 0 length String the property is added. If a property is not registered and the property value is a 0 length String the property is ignored.

**Parameters:**
- `properties` - The set of properties to be registered.
- `persist` - If true the properties parameters are stored in persistent storage, otherwise they are not stored and SHALL be removed immediately if previously stored.
- `expirationDate` - Date the implementation SHALL remove the properties from persistent storage.

**Throws:**
- `java.lang.SecurityException` - it the calling application is not granted MonitorAppPermission("properties").

**getAddressingProperties**

```java
public java.util.Properties getAddressingProperties()
```

Gets the addressing properties previously registered by the registerAddressingProperties method. The set of properties returned by this method may be out of date as soon as this method returns.
Returns:
The set of registered addressing properties. If no addressing properties have been registered an empty
Properties object is returned.

removeAddressingProperties

public void removeAddressingProperties(java.lang.String[] properties)

Removes addressing properties set by the registerAddressingProperties method. Each String in
the properties parameter SHALL be compared to registered property keys and if a match is found the
property SHALL be removed. If the properties parameter is null all registered properties SHALL be
removed from both volatile storage and non-volatile storage if persistently stored.
Parameters:
properties - The properties to remove.

Throws:
java.lang.SecurityException - if the calling application is not granted
Monitor.AppPermission("properties").

getSecurityAddressableAttributes

public java.util.Properties getSecurityAddressableAttributes()

Gets the security system Host addressable attributes queried by the implementation. The implementation
SHALL format addressable attributes sent by the security system into name/value pairs in the returned
Properties. The set of properties returned by this method may be out of date as soon as this method
returns.

Returns:
The set of addressable attributes set by the security system.
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.
- 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 1.0.2 [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

<table>
<thead>
<tr>
<th>static int</th>
<th>ALLOW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When AppFilter.accept finds a matching AppPattern with this action, it returns true.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>ASK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When AppFilter.accept finds a matching AppPattern with this action, it asks AppFilterHandler.accept for the decision.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>DENY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When AppFilter.accept finds a matching AppPattern with this action, it returns false.</td>
</tr>
</tbody>
</table>

Constructor Summary

**AppPattern**(java.lang.String idPattern, int action, int priority)
Constructs a new AppPattern with no expiration.

**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.

Method Summary

<table>
<thead>
<tr>
<th>boolean equals(java.lang.Object that)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indicates whether some other object is &quot;equal to&quot; this one.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int getAction()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the action associated with this AppPattern.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>java.lang.String getAppIDPattern()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the pattern string that specifies a group of AppIDs.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>java.util.Date getExpirationTime()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the time for this AppPattern to expire or null if it never expires.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int getPriority()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the search order priority of this AppPattern.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>java.lang.Object getPrivateInfo()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns MSO-private information of this AppPattern.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int hashCode()</th>
</tr>
</thead>
</table>

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), Constant Field Values

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), Constant Field Values

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), Constant Field Values

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

This interface represents a handler that can be registered in order to receive a notification whenever the XAIT version changes. An application which has a MonitorAppPermission("registrar") may have a class implementing this interface, and may set it as the AppSignalHandler using the AppManagerProxy.

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean notifyXAITUpdate</td>
<td>Notifies the registered handler that the version of the signaled XAIT has changed.</td>
</tr>
</tbody>
</table>

Method Detail

notifyXAITUpdate

boolean notifyXAITUpdate(OcapAppAttributes[] newApps)

Notifies the registered handler that the version of the signaled XAIT has changed. When an AppSignalHandler is registered the implementation SHALL call this method whenever the version of the XAIT changes or when an XAIT is received and a previous call to this method for the same version of the XAIT returned false.

When the return value is true, the implementation SHALL process the corresponding XAIT as required by the OCAP specification. When the return value is false, the implementation SHALL ignore the corresponding XAIT and treat it as if it were never received. Subsequent XAIT receipt with the same version as the ignored table is treated as a new version and notified using this method.

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.

Returns:
- When true, the XAIT is processed; otherwise, the XAIT is ignored by the implementation.
org.ocap.application
Interface OcapAppAttributes

All Superinterfaces:
    AppAttributes

public interface OcapAppAttributes
extends 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.

Field Summary

<table>
<thead>
<tr>
<th>Static</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>AUTOSTART</td>
<td>This represents the application control code &quot;AUTOSTART&quot; defined for the application_control_code in an AIT or a XAIT.</td>
</tr>
<tr>
<td>int</td>
<td>BACKGROUND_MODE</td>
<td>This represents the &quot;background&quot; application mode defined for the application_mode_descriptor in an AIT or XAIT.</td>
</tr>
<tr>
<td>int</td>
<td>CROSSENVIRONMENT_MODE</td>
<td>This represents the &quot;cross-environment&quot; application mode defined for the application_mode_descriptor in an AIT or XAIT.</td>
</tr>
<tr>
<td>int</td>
<td>DESTROY</td>
<td>This represents the application control code &quot;DESTROY&quot; defined for the application_control_code in an AIT or a XAIT.</td>
</tr>
<tr>
<td>int</td>
<td>KILL</td>
<td>This represents the application control code &quot;KILL&quot; defined for the application_control_code in an AIT or a XAIT.</td>
</tr>
<tr>
<td>int</td>
<td>LEGACY_MODE</td>
<td>This represents the &quot;legacy&quot; application mode defined for the application_mode_descriptor in an AIT or XAIT.</td>
</tr>
<tr>
<td>int</td>
<td>NORMAL_MODE</td>
<td>This represents the &quot;normal&quot; application mode defined for the application_mode_descriptor in an AIT or XAIT.</td>
</tr>
<tr>
<td>int</td>
<td>OCAP_J</td>
<td>The OCAP registered value for all OCAP-J applications.</td>
</tr>
</tbody>
</table>
Field Summary

<table>
<thead>
<tr>
<th>static int</th>
<th>PAUSED_MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This represents the &quot;paused&quot; application mode defined for the application_mode_descriptor in an AIT or XAIT.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>PREFETCH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The application control code &quot;PREFETCH&quot; is only used for DVB-HTML applications.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>PRESENT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This represents the application control code &quot;PRESENT&quot; defined for the application_control_code in an AIT or a XAIT.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>REMOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This represents the application control code &quot;REMOTE&quot; defined for the application_control_code in an AIT.</td>
</tr>
</tbody>
</table>

Fields inherited from interface org.dvb.application.AppAttributes

DVB_HTML_application, DVB_J_application

Method Summary

<table>
<thead>
<tr>
<th>int</th>
<th>getApplicationControlCode()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This method returns the application_control_code of the application represented by this interface.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getApplicationMode()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the signaled application mode for this application.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getStoragePriority()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This method returns the currently set storage priority for the application.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>hasNewVersion()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Indicates that a new version of the application is stored that will replace the currently launched version when a new lifecycle for this application starts.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>isNewVersionSignaled()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Indicates that a new version of the application is available and will replace the currently launched version when a new lifecycle for this application starts.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>static final int</th>
<th>OCAP_J</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The OCAP registered value for all OCAP-J applications.</td>
</tr>
</tbody>
</table>

See Also:

Constant Field Values
AUTOSTART
static final int AUTOSTART
    This represents the application control code "AUTOSTART" defined for the application_control_code in an AIT or a XAIT.
    See Also:
    Constant Field Values

PRESENT
static final int PRESENT
    This represents the application control code "PRESENT" defined for the application_control_code in an AIT or a XAIT.
    See Also:
    Constant Field Values

DESTROY
static final int DESTROY
    This represents the application control code "DESTROY" defined for the application_control_code in an AIT or a XAIT.
    See Also:
    Constant Field Values

KILL
static final int KILL
    This represents the application control code "KILL" defined for the application_control_code in an AIT or a XAIT.
    See Also:
    Constant Field Values

PREFETCH
static final int PREFETCH
    The application control code "PREFETCH" is only used for DVB-HTML applications.
    See Also:
    Constant Field Values

REMOTE
static final int REMOTE
    This represents the application control code "REMOTE" defined for the application_control_code in an AIT.
    See Also:
    Constant Field Values

LEGACY_MODE
static final int LEGACY_MODE
    This represents the "legacy" application mode defined for the application_mode_descriptor in an AIT or XAIT.
    See Also:
    Constant Field Values
NORMAL_MODE
static final int NORMAL_MODE
  This represents the "normal" application mode defined for the application_mode_descriptor in an AIT or XAIT.
  See Also:
  Constant Field Values

CROSSENVIRONMENT_MODE
static final int CROSSENVIRONMENT_MODE
  This represents the "cross-environment" application mode defined for the application_mode_descriptor in an or XAIT.
  See Also:
  Constant Field Values

BACKGROUND_MODE
static final int BACKGROUND_MODE
  This represents the "background" application mode defined for the application_mode_descriptor in an AIT or XAIT.
  See Also:
  Constant Field Values

PAUSED_MODE
static final int PAUSED_MODE
  This represents the "paused" application mode defined for the application_mode_descriptor in an AIT or XAIT.
  See Also:
  Constant Field Values

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 that will replace the currently launched version when a new lifecycle for this application starts.
This method SHALL return true if \texttt{getStoragePriority()} would return non-zero for the newer version of the application after it replaces the currently launched version. This method SHALL return false if \texttt{isNewVersionSignaled()} would return false.

\textbf{Returns:}
True, if the application is currently launched but a new version is signaled and stored. False, if the application is not currently launched, if the currently signaled version matches the currently launched version, or the new version is not yet stored.

\textbf{isNewVersionSignaled}

\texttt{boolean isNewVersionSignaled()}

Indicates that a new version of the application is available and will replace the currently launched version when a new lifecycle for this application starts.

\textbf{Returns:}
True, if the application is currently launched but a new version is signaled. False, if the application is not currently launched or if the currently signaled version matches the currently launched version.

\textbf{getApplicationMode}

\texttt{int getApplicationMode()}

Returns the signaled application mode for this application. That is, the application mode that this application would run in when the application's home environment is not \texttt{selected} or \texttt{presenting}.

A value of \texttt{NORMAL\_MODE} indicates that the application will be terminated when the home environment is not selected or presenting. Any other value indicates the mode that the application would run in.

Where no \texttt{application\_mode\_descriptor} was signaled, the default value of \texttt{LEGACY\_MODE} SHALL be returned.

Where the host does not support the signaled \texttt{application\_mode\_descriptor} and it is therefore otherwise ignored, the int corresponding to the signaled \texttt{application\_mode\_descriptor} SHALL be returned as specified in the return values listed below.

The current application mode can be determined by consulting \texttt{Environment.getState()} in addition to considering the signaled application mode.

\textbf{Returns:}
one of \texttt{LEGACY\_MODE}, \texttt{NORMAL\_MODE}, \texttt{CROSSENVIRONMENT\_MODE}, \texttt{BACKGROUND\_MODE}, or \texttt{PAUSED\_MODE}.
org.ocap.application
Class OcapIxcPermission

java.lang.Object
   java.security.Permission
      java.security.BasicPermission
         org.ocap.application.OcapIxcPermission

All Implemented Interfaces:
   java.io.Serializable, java.security.Guard

public final class OcapIxcPermission
extends java.security.BasicPermission

This class represents access to the inter-xlet communication registry. An OcapIxcPermission consists of a name specification and an action specifying what can be done with those names.

The name specification is a superset of the name passed into the IxcRegistry methods such as IxcRegistry.bind(javax.tv.xlet.XletContext, java.lang.String, java.rmi.Remote) and IxcRegistry.lookup(javax.tv.xlet.XletContext, java.lang.String). Valid names are composed of fields delimited by "/" characters, with each field specifying a particular value (e.g., OID). The following grammar defines the name format:

NAME = "+" | "/" SCOPE "+" SIGNED "+" OID "+" AID "+" BINDNAME
SCOPE = "+" | "global" | "ixc" | "service-" CONTEXT
CONTEXT = "+" | context-id
SIGNED = "+" | "signed" | "unsigned"
OID = "+" | oid
AID = "+" | aid
BINDNAME = "+" | bindname | bindname "+"

Where "+" specifies a wildcard character. Where context-id is a platform-specific unique identifier for a service context; oid and aid are the organization and application identifiers of the binding application as converted by Integer.toHexString(int); and bindname is the application-defined name given at bind-time.

- "+*" as the entire name string will match any other name
- "*//*/*/*/*" is equivalent to "+*
- "/*/1a/4abc/" will match names in any scope, published by an application with an OID of 1a and AID of 4abc.
- "/*/signed/*/VODApi" will match any object bound by a signed application with an ixcname of "VODApi".

The actions specification is comprised of a single action specified by one of two keywords: "bind" or "lookup". These correspond to the bind and lookup methods of IxcRegistry. The actions string is converted to lowercase before processing.

See Also:
   Serialized Form
Constructor Summary

OcapIxcPermission(java.lang.String name, java.lang.String actions)
Creates a new OcapIxcPermission object with the specified name and actions.

Method Summary

<table>
<thead>
<tr>
<th>boolean equals(java.lang.Object obj)</th>
<th>Checks two OcapIxcPermission objects for equality.</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.lang.String getActions()</td>
<td>Returns the &quot;canonical string representation&quot; of the actions.</td>
</tr>
<tr>
<td>int hashCode()</td>
<td>Returns the hash code value for this object.</td>
</tr>
<tr>
<td>boolean implies(java.security.Permission p)</td>
<td>Checks if this OcapIxcPermission &quot;implies&quot; the specified permission.</td>
</tr>
<tr>
<td>java.security.PermissionCollection newPermissionCollection()</td>
<td>Returns a new PermissionCollection object for storing OcapIxcPermission objects.</td>
</tr>
</tbody>
</table>

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

OcapIxcPermission

public OcapIxcPermission(java.lang.String name, java.lang.String actions)
Creates a new OcapIxcPermission object with the specified name and actions. The name specification is a superset of the name passed into the IxcRegistry methods such as IxcRegistry.bind(javax.tv.xlet.XletContext, java.lang.String, java.rmi.Remote) and IxcRegistry.lookup(javax.tv.xlet.XletContext, java.lang.String). See the class description for the specification of the name string.

The actions specification is comprised of a single action specified by one of two keywords: "bind" or "lookup". These correspond to the bind and lookup methods of IxcRegistry. The actions string is converted to lowercase before processing.

Parameters:
name - The name specification for exported/imported objects
actions - The action string
Method Detail

equals

public boolean equals(java.lang.Object obj)
    Checks two OcapIxcPermission objects for equality. Check that other is an OcapIxcPermission, and has the
    same name and actions as this object.
    Overrides:
equals in class java.security.BasicPermission
Parameters:
    obj - the object we are testing for equality with this object
Returns:
    true if obj is an OcapIxcPermission, and has the same name and actions as this OcapIxcPermission object.

getActions

public java.lang.String getActions()
    Returns the "canonical string representation" of the actions. That is, this method always returns present
    actions in the following order: bind, lookup. For example, if this OcapIxcPermission object allows both
    bind and lookup actions, a call to getActions will return the string "bind,lookup".
    Overrides:
    getActions in class java.security.BasicPermission
Returns:
    the canonical string representation of the actions

hashCode

public int hashCode()
    Returns the hash code value for this object.
    Overrides:
    hashCode in class java.security.BasicPermission
Returns:
    a hash code value for this object.

implies

public boolean implies(java.security.Permission p)
    Checks if this OcapIxcPermission "implies" the specified permission.
    More specifically, this method returns true if:
    - p is an instanceof OcapIxcPermission
    - p's actions are a proper subset of this object's actions, and
    - p's name is implied by this object's name.
    The rules for determining if this object's name implies p's name are as follows:
    - Where p's name is exactly the same as this object's name, then it is implied.
    - The name "/*" and "/*//*/*//*/*" both imply all possible names.
    - Where this object's name includes a wildcard for a field ("*"), then all possible values for that field
      are implied.
Where this object's name includes a field that ends in a wildcard (e.g., service-*) then all possible values for that field starting with the non-wildcard portion are implied.

For example, "/service-*/signed/abc/4001/*" implies "/service-1234/signed/abc/4001/VODObject".

An OcapIxcPermission may also imply an IxcPermission. That is, this method will also return true if:

- p is an instanceof IxcPermission
- p's actions are a proper subset of this object's actions, and
- p's name is implied by this object's name.

The rules for determining if this object's name implies an IxcPermission name are the same as detailed above except that a translation of the IxcPermission name to the OcapIxcPermission is applied first. The following table shows how such a mapping SHALL be applied:

<table>
<thead>
<tr>
<th>IxcPermission name</th>
<th>OcapIxcPermission name</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;dvb://<em>/</em>&quot;</td>
<td>&quot;global/signed/<em>/</em>/*&quot;</td>
</tr>
<tr>
<td>&quot;dvb:/signed/*&quot;</td>
<td>&quot;service-id/signed/<em>/</em>/*&quot;</td>
</tr>
<tr>
<td>&quot;dvb:/service/id/signed/*&quot;</td>
<td>&quot;service-id/signed/<em>/</em>/*&quot;</td>
</tr>
<tr>
<td>&quot;dvb:/ixc/*&quot;</td>
<td>&quot;ixc/<em>/</em>/<em>/</em>&quot;</td>
</tr>
<tr>
<td>&quot;dvb:/signed/_OID/&quot;</td>
<td>&quot;global/signed/_OID/*&quot;</td>
</tr>
<tr>
<td>&quot;dvb:/service/id/signed/_OID/*&quot;</td>
<td>&quot;service-id/signed/_OID/*&quot;</td>
</tr>
<tr>
<td>&quot;dvb:/ixc/_OID/&quot;</td>
<td>&quot;ixc/<em>/_OID/</em>&quot;</td>
</tr>
<tr>
<td>&quot;dvb:/signed/_OID/_AID/&quot;</td>
<td>&quot;global/signed/_OID/&quot;</td>
</tr>
<tr>
<td>&quot;dvb:/service/id/signed/_OID/&quot;</td>
<td>&quot;service-id/signed/_OID/&quot;</td>
</tr>
<tr>
<td>&quot;dvb:/ixc/_OID/&quot;</td>
<td>&quot;ixc/<em>/_OID/</em>&quot;</td>
</tr>
<tr>
<td>&quot;dvb:/signed/_OID/&quot;</td>
<td>&quot;global/signed/_OID/*&quot;</td>
</tr>
<tr>
<td>&quot;dvb:/service/id/signed/_OID/&quot;</td>
<td>&quot;service-id/signed/_OID/&quot;</td>
</tr>
<tr>
<td>&quot;dvb:/ixc/_OID/&quot;</td>
<td>&quot;ixc/<em>/_OID/</em>&quot;</td>
</tr>
</tbody>
</table>

Any IxcPermission name that cannot be mapped cannot be implied.

Overrides:
implies in class java.security.BasicPermission
Parameters:
p - the permission to check against
Returns:
true if the specified permission is implied by this object, false if not.

**newPermissionCollection**

public java.security.PermissionCollection newPermissionCollection()

Returns a new PermissionCollection object for storing OcapIxcPermission objects.
OcapIxcPermission objects must be stored in a manner that allows them to be inserted into the collection in any order, but that also enables the PermissionCollection implies method to be implemented in an efficient (and consistent) manner.

**Overrides:**
newPermissionCollection in class java.security.BasicPermission

**Returns:**
a new PermissionCollection object suitable for storing OcapIxcPermissions.
**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

<table>
<thead>
<tr>
<th>protected PermissionInformation()</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCAP applications SHALL NOT use this constructor - it is provided for internal use by the OCAP implementation.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>abstract AppID getAppID()</th>
</tr>
</thead>
<tbody>
<tr>
<td>This method returns an AppID of an application to be granted a requested set of Permissions that is returned by the getRequestedPermissions() method.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>abstract java.security.cert.Certificate[][] getCertificates()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the set of valid certificates that were used to sign the application identified by the AppID returned by the getAppID() method.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>abstract java.security.PermissionCollection getRequestedPermissions()</th>
</tr>
</thead>
<tbody>
<tr>
<td>This method returns the requested set of Permissions for the application specified by the AppID that is returned by the getAppID() method.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static java.security.PermissionCollection getUnsignedAppPermissions()</th>
</tr>
</thead>
<tbody>
<tr>
<td>This method returns the set of Permissions that are requested by all unsigned applications.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>abstract boolean isManufacturerApp()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns true if and only if the application identified by the AppID returned by the getAppID() is a Host Device Manufacturer applications.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>abstract boolean isPrivilegedCertificate(java.security.cert.Certificate cert)</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>
### Constructor Detail

#### PermissionInformation

```java
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

```java
public abstract 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

```java
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 application.

**Returns:**
true if and only if the application identified by the AppID returned by the `getAppID()` is a Host Device Manufacturer application.

#### getCertificates

```java
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**

```java
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**

```java
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**

```java
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(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(SecurityPolicyHandler)

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>java.security.PermissionCollection getAppPermissions(PermissionInformation permissionInfo)</code></td>
<td>This callback method is used to modify the set of Permissions that is granted to an application to be launched.</td>
</tr>
</tbody>
</table>

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 MPEG Component API

This annex defines the org.ocap.mpeg APIs.

Table H–1 - Correlation between OCAP and [DVB-GEM 1.0.2]

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annex H, OCAP MPEG Component API</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
</tbody>
</table>

Package org.ocap.mpeg

The org.ocap.mpeg package provides access to private data sections in the extended channel

See:

Description

Class Summary

<table>
<thead>
<tr>
<th>Class Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PODExtendedChannel</td>
<td>This class represents an extended channel that provides access to private section data flows.</td>
</tr>
</tbody>
</table>

Package org.ocap.mpeg Description

The org.ocap.mpeg package provides access to private data sections in the extended channel
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**

<table>
<thead>
<tr>
<th>protected PODExtendedChannel()</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCAP applications SHALL NOT use this method - it is provided for internal use by the OCAP implementation.</td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>static PODExtendedChannel getInstance()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets a PODExtendedChannel instance.</td>
</tr>
</tbody>
</table>

**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

PODEntendedChannel

protected PODEntendedChannel()

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 PODEntendedChannel getInstance()

Gets a PODEntendedChannel instance. The implementation MAY return the same instance each time, or it MAY return different (but functionally identical) instances.

Returns:
A PODEntendedChannel instance.
Annex I  OCAP Net API

Table I–1 - Correlation between OCAP and [DVB-GEM 1.0.2]

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annex I OCAP Net API</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
</tbody>
</table>

Package org.ocap.net

The org.ocap.net package provides locators for accessing various types of items in the transport stream.

See:

Description

Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OcapLocator</td>
<td>This class encapsulates an OCAP URL into an object.</td>
</tr>
<tr>
<td>OCRCInterface</td>
<td>This class models a return channel interface for use in receiving and transmitting IP packets over an OCAP-compliant return channel.</td>
</tr>
<tr>
<td>URLLocator</td>
<td>A concrete implementation of Locator that encapsulates a URL into an object.</td>
</tr>
</tbody>
</table>

Package org.ocap.net Description

The org.ocap.net package provides locators for accessing various types of items in the transport stream.
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 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.

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

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OcapLocator(int sourceID)</td>
<td>A constructor of this class corresponding to the OCAP URL form &quot;ocap://source_id&quot;.</td>
</tr>
<tr>
<td>OcapLocator(int frequency, int modulationFormat)</td>
<td>A constructor of this class corresponding to the OCAP URL form &quot;ocap://f=frequency[m=modulation_format]&quot;.</td>
</tr>
<tr>
<td>OcapLocator(int sourceID, int[] PID, int eventID, java.lang.String pathSegments)</td>
<td>A constructor of this class corresponding to the OCAP URL form &quot;ocap://source_id[.eventID[.PID[pathSegments]]].&quot;</td>
</tr>
</tbody>
</table>
Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OcapLocator(int frequency, int programNumber, int modulationFormat)</code></td>
<td>A constructor of this class corresponding to the OCAP URL form &quot;ocap://(oobfdc</td>
</tr>
<tr>
<td><code>OcapLocator(int sourceID, int eventID, int[] componentTags, java.lang.String pathSegments)</code></td>
<td>A constructor of this class corresponding to the OCAP URL form &quot;ocap://source_id&quot;.</td>
</tr>
<tr>
<td><code>OcapLocator(int frequency, int programNumber, int modulationFormat, int[] PID, int eventID, java.lang.String pathSegments)</code></td>
<td>A constructor of this class corresponding to the OCAP URL form &quot;ocap://(oobfdc</td>
</tr>
<tr>
<td><code>OcapLocator(int frequency, int programNumber, int modulationFormat, int eventID, java.lang.String pathSegments)</code></td>
<td>A constructor of this class corresponding to the OCAP URL form &quot;ocap://(oobfdc</td>
</tr>
<tr>
<td><code>OcapLocator(int frequency, int programNumber, int modulationFormat, short[] streamType, int[] index, int eventID, java.lang.String pathSegments)</code></td>
<td>A constructor of this class corresponding to the OCAP URL form &quot;ocap://(oobfdc</td>
</tr>
<tr>
<td><code>OcapLocator(int frequency, int programNumber, int modulationFormat, java.lang.String[] ISO639LanguageCode, int eventID, java.lang.String pathSegments)</code></td>
<td>A constructor of this class corresponding to the OCAP URL form &quot;ocap://(oobfdc</td>
</tr>
<tr>
<td><code>OcapLocator(int sourceID, short[] streamType, java.lang.String[] ISO639LanguageCode, int eventID, java.lang.String pathSegments)</code></td>
<td>A constructor of this class corresponding to the OCAP URL form &quot;ocap://source_id[stream_type[index][ISO_639_language_code][eventID]/path_segments]&quot;.</td>
</tr>
<tr>
<td><code>OcapLocator(int sourceID, java.lang.String[] componentName, int eventID, java.lang.String pathSegments)</code></td>
<td>A constructor of this class corresponding to the OCAP URL form &quot;ocap://source_id[componentName][eventID]/path_segments]&quot;.</td>
</tr>
<tr>
<td><code>OcapLocator(java.lang.String url)</code></td>
<td>A constructor of this class for any form of OCAP URL.</td>
</tr>
<tr>
<td><code>OcapLocator(java.lang.String serviceName, int[] PID, int eventID, java.lang.String pathSegments)</code></td>
<td>A constructor of this class corresponding to the OCAP URL form &quot;ocap://n=service_name&quot;.</td>
</tr>
</tbody>
</table>
Constructor Summary

**OcapLocator***(java.lang.String serviceName, int eventID, int[] componentTags, java.lang.String pathSegments)*

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)*

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)*

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)*

A constructor of this class corresponding to the OCAP URL form "ocap://n=service_name[.

Method Summary

**java.lang.String[]** getComponentNames()  
This method returns a component_name value of the OCAP URL represented by this OcapLocator instance.

**int[]** getComponentTags()  
This method returns a component_tag value of the OCAP URL represented by this OcapLocator instance.

**int** getEventId()  
This method returns an event_id value of the OCAP URL represented by this OcapLocator instance.

**int** getFrequency()  
This method returns a frequency value, in hertz, of the OCAP URL represented by this OcapLocator instance.

**int[]** getIndexes()  
This method returns an index value of the OCAP URL represented by this OcapLocator instance.

**java.lang.String[]** getLanguageCodes()  
This method returns an ISO_639_language_code value of the OCAP URL represented by this OcapLocator instance.

**int** getModulationFormat()  
This method returns a value representing a modulation_format as specified in SCTE 65.

**java.lang.String** getPathSegments()  
This method returns a path_segments string of the OCAP URL represented by this OcapLocator instance.

**int[]** getPIDs()  
This method returns a PID value of the OCAP URL represented by this OcapLocator instance.
### Method Summary

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>getProgramNumber()</td>
<td>This method returns a program_number value of the OCAP URL represented by this OcapLocator instance.</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>getServiceName()</td>
<td>This method returns a service_name value of the OCAP URL represented by this OcapLocator instance.</td>
</tr>
<tr>
<td>int</td>
<td>getSourceID()</td>
<td>This method returns a source_id value of the OCAP URL represented by this OcapLocator instance.</td>
</tr>
<tr>
<td>short[]</td>
<td>getStreamTypes()</td>
<td>This method returns a stream_type value of the OCAP URL represented by this OcapLocator instance.</td>
</tr>
</tbody>
</table>

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

Methods inherited from interface javax.tv.locator.Locator

- equals, hashCode

### Constructor Detail

**OcapLocator**

```java
public OcapLocator(int sourceID) throws 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:**

- InvalidLocatorException - if the sourceID to construct the locator doesn't specify a valid OCAP URL (e.g., a value is out of range).

**OcapLocator**

```java
public OcapLocator(int frequency,
                   int programNumber,
                   int modulationFormat)
```

throws 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 
0xFF the modulation_format is treated as NTSC analog and the programNumber parameter is ignored. 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:
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 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 
0xFF the modulation_format is treated as NTSC analog. 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:
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 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:
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 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:**
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 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 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 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:
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 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:
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 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:
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 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:
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 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:
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 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:
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 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:
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 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:
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 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:
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 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 0xFF the modulation_format is treated as NTSC analog and the programNumber parameter is ignored. 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:
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 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
0xFF the modulation_format is treated as NTSC analog and the programNumber parameter is ignored. 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:
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 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 "oobfide" 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 0xFF the modulation_format is treated as NTSC analog and the programNumber parameter is ignored. 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:**

- `InvalidLocatorException` - if the parameters to construct the locator don't specify a valid OCAP URL (e.g., a value is out of range).

```java
public OcapLocator(int frequency,
                    int programNumber,
                    int modulationFormat,
                    java.lang.String[] componentName,
                    int eventId,
                    java.lang.String pathSegments)
    throws InvalidLocatorException
```

A constructor of this class corresponding to the OCAP URL form "ocap://(oobfide|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 "oobfide" 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 0xFF the modulation_format is treated as NTSC analog and the programNumber parameter is ignored. 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:**

- `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 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 event Id 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
0xFF the modulation_format is treated as NTSC analog and the programNumber parameter is ignored. 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:
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

ggetSourceID

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.

ggetServiceName

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**

```java
def 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**

```java
def getModulationFormat()
```

This method returns a value representing a modulation_format as specified in SCTE 65. A modulation_format value of 0xFF indicates an NTSC analog video format.

**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**

```java
def 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**

```java
def 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**

```java
def 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.
Class OCRCInterface

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 | Constant to indicate a DOCSIS return channel. |
| static int | SUBTYPE_CATV_OOB    | Constant to indicate an OOB return channel, either SCTE 55-2 or SCTE 55-1, accessed through the CableCARD interface as specified in CCIF 2.0. |

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() 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

close, 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.

See Also:

Constant Field Values

**SUBTYPE_CATV_OOB**

public static final int SUBTYPE_CATV_OOB

Constant to indicate an OOB return channel, either SCTE 55-2 or SCTE 55-1, accessed through the CableCARD interface as specified in CCIF 2.0.

See Also:

Constant Field Values

Constructor Detail

**OCRCInterface**

protected OCRCInterface()
org.ocap.net
Class URLLocator

java.lang.Object
   org.davic.net.Locator
      org.ocap.net.URLLocator

All Implemented Interfaces:
   javax.tv.locator.Locator

public class URLLocator
extends Locator

A concrete implementation of Locator that encapsulates a URL into an object.

Instances of this class MAY be used to reference any resource that is referencable using an instance of URL.

See Also:
   URL

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URLLocator(java.lang.String url)</td>
<td>Construct an URLLocator encapsulating the given URL.</td>
</tr>
<tr>
<td>URLLocator(java.net.URL url)</td>
<td>Construct an URLLocator encapsulating the given URL.</td>
</tr>
</tbody>
</table>

### Method Summary

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

URLLocator

public URLLocator(java.lang.String url)
throws InvalidLocatorException

Construct an URLLocator encapsulating the given URL.

The accepted syntax SHALL be identical to that accepted by URL.URL(String).
Parameters:
url - a URL string

Throws:
InvalidLocatorException - if URL(String) would throw MalformedURLException

URLLocator

public URLLocator(java.net.URL url)

Construct an URLLocator encapsulating the given URL.

This SHALL be equivalent to new URLLocator(url.toString()).

Parameters:
url - the URL expressed as an instance of URL

See Also:
URLLocator(String)
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

Annex J (this section) of OCAP corresponds to Annex Q of [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2] as follows:

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2]</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3]</th>
<th>MHP Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annex J.1 DVB-GEM and DVB-MHP Specification Correspondence</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>Annex J.2 OCAP Specific Requirements</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>Annex J.2.1 Extensions to DVB/MHP (Normative)</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
<tr>
<td>Annex J.2.1.1 Package javax.tv.net Description</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td></td>
</tr>
</tbody>
</table>

The OpenCable Application Platform is in complete compliance with the API, org.dvb.net, specified in this section.

J.2 OCAP Specific Requirements

J.2.1 Extensions to DVB/MHP (Normative)

This information extends the specification requirements made to the [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2].

The following information is normative for OCAP. It is specific to OCAP and is not contained within the [DVB-MHP 1.0.3] or [DVB-GEM 1.0.2].

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 using the out-of-band return channel for transmission of IP datagrams.
Annex K  OCAP 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 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:

1. Modify the event.
2. 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

Annex K (this annex) of OCAP corresponds to Annex J of [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2] as follows:

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>MHP Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annex K.1 DVB-GEM and DVB-MHP Specification Correspondence</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
<tr>
<td>Annex K.2 OCAP Specific Requirements</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
</tbody>
</table>

K.2 OCAP Specific Requirements

This section extends Annex J of [DVB-GEM 1.0.2]. 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:
Start of OCAP downloaded application visible event processing

Does the event contain Mandatory Ordinary Keycode?

Filter the event by UserEventFilter

Yes

No

Does a downloaded application have focus, or is the event in a repository registered in the EventManager?

Yes

No

Does a an application have this event reserved exclusively?

Yes

No

Does a an application have AWT focus?

Yes

Is this event in the set of key events which this HScene is interested in receiving?

Yes

No

Send to AWT application with focus via AWT Event Queue

No

Do any applications have this event registered in an EventRepository for shared access?

Yes

Send to applications via the org.dvb.event.EventListener

No

End of OCAP downloaded application visible event processing

Figure K–1 - The OCAP downloaded application visible event distribution mechanism

The above diagram illustrates event propagation when cable is the selected environment. When cable is not selected, the policy of the selected environment may impact an application's ability to gain focus or retain event reservations.
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

```java
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 in Multi-function implementations

This section describes key event processing in multi-function implementations.

The rules and policies of the selected environment govern input event reservations and propagation. As such, when
cable is not the selected environment, cable applications MAY lose input event reservations. Cable applications
SHALL be made aware of these conditions via the DAVIC resource notification API; applications MAY re-establish
their reservations when cable becomes selected. When cable applications have input event reservations, or cross-
environment cable applications have input focus, the implementation SHALL adhere to the OCAP event distribution
model outlined in Annex K.2.2.
K.2.6 org.ocap.event API

Package org.ocap.event

The org.ocap.event package manages user events.

See:
  Description

<table>
<thead>
<tr>
<th>Interface Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserEventFilter</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventManager</td>
</tr>
<tr>
<td>UserEvent</td>
</tr>
<tr>
<td>UserEventAction</td>
</tr>
</tbody>
</table>

Package org.ocap.event Description

The org.ocap.event package manages user events.
org.ocap.event
Class EventManager

java.lang.Object
   org.dvb.event.EventManager
   org.ocap.event.EventManager

All Implemented Interfaces:
   ResourceServer

public class EventManager
   extends 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() |
| Construct for an instance of this class. |

Method Summary

| UserEventRepository | getFilteredRepository() |
| Get the current UserEventRepository which specify the events to be filtered. |

| static EventManager | getInstance() |
| This method returns the sole instance of the org.ocap.event.EventManager class. |

| void setFilteredRepository(UserEventRepository repository) |
| Sets the repository which specifies the events to be filtered. |
Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void setUserEventFilter(UserEventFilter filter)</td>
<td>Set the specified UserEventFilter to modify or consume the event and/or change the applications to deliver the event to.</td>
</tr>
</tbody>
</table>

Methods inherited from class org.dvb.event.EventManager

- addExclusiveAccessToAWTEvent
- addResourceStatusEventListener
- 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 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 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(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 result in an update of the UserEventRepository, 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**

```java
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 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.

See Also:
  Serialized Form

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)
  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)
  Constructor for a new UserEvent object representing a key being pressed.

Method Summary

void setCode(int code)
  Modifies the event code.

void setKeyChar(char keychar)
  Modifies the character associated with the key in this event.
### Constructor Detail

**UserEvent**

```java
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**

```java
public UserEvent(java.lang.Object source,
                 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, AppID[] appIDs)
    Creates a UserEventAction instance.

Method Summary

<table>
<thead>
<tr>
<th>AppID[]</th>
<th>getAppIDs()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Get the AppIDs to which the filtered event will be forwarded.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UserEvent</th>
<th>getEvent()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Get the event to be forwarded.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setAppIDs(AppID[] appIDs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets the application IDs returned by this class.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setEvent(UserEvent event)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets the event returned by this class.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
close, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

public UserEventAction(UserEvent event, 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**

```java
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**

```java
public void setAppIDs(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**

```java
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**

```java
public 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

<table>
<thead>
<tr>
<th>UserEventAction</th>
<th>filterUserEvent(UserEvent e)</th>
</tr>
</thead>
</table>
|                   | 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 Resource Management API

This section presents the `org.ocap.resource` APIs.

### Table L–1 - Correlation between OCAP and [DVB-GEM 1.0.2]

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annex L OCAP Resource Management API</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
</tbody>
</table>

### Package 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.

See:

- Description

#### Interface Summary

- **ApplicationResourceUsage**
  
  This interface represents a ResourceUsage corresponding to a resource explicitly reserved by an application by 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.

- **SharedResourceUsage**
  
  This interface represents a group of resources where one or more resources are shared between multiple resource usages.

#### Class Summary

- **ResourceContentionManager**
  
  This class manages a means of resolving a resource contention.

### Package org.ocap.resource Description

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 assigns priority based upon expressed need, allowing higher priority for resource reservation to an application that has an organization ID of `PRIORITIZED_ORGANIZATION`.

```java
import org.ocap.resource.*;
import org.dvb.application.*;

public class ResourceHandler extends AppsDatabaseFilter
    implements ResourceContentionHandler, ResourceContentionHandler2 {

    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() {
        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) {
        return appid.getOID() != REJECTED_ORGANIZATION;
    }

    /*
     * Resolve a resource contention.
     */
    public ResourceUsage[] resolveResourceContention(
        ResourceUsage[] newRequests,
        ResourceUsage[] currentReservations) {
        // Copy new requests and current reservations into single array
        ResourceUsage[] result =
            new ResourceUsage[newRequests.length +
                currentReservations.length];
        System.arraycopy(newRequests, 0,
            result, 0,
            newRequests.length);
        System.arraycopy(currentReservations, 0,
            result, newRequests.length,
            currentReservations.length);
```

The code provided is a sample implementation of a resource handling mechanism in an OCAP monitor application. It demonstrates how to check for application organization IDs, and to handle resource reservations and contentions using the `ResourceContentionHandler` interface. This example is part of the implementation of the OCAP system to manage resources efficiently among various applications.
currentReservations.length);

    // Use bubble sort to order all requests by priority descending
    for(int i = 0; i < result.length; i++)
    {
        for(int j = 0; j < result.length - 1; j++)
        {
            if (mapPriority(result[j]) < mapPriority(result[j+1]))
            {
                ResourceUsage temp = result[j];
                result[j] = result[j+1];
                result[j+1] = temp;
            }
        }
    }

    // Return array contain all new requests and current reservations
    // ordered by priority descending.
    return result;

} /*
 * Resolve a resource contention.
 */
public ResourceUsage[] resolveResourceContention(
    ResourceUsage newRequest,
    ResourceUsage[] currentReservations)
{
    return resolveResourceContention(new ResourceUsage[] { newRequest },
                                       currentReservations);
}

/*
 * Map given ResourceUsage to desired priority.
 */
private static int mapPriority(ResourceUsage ru) {
    if (ru == null)
        return -1;
    int p = ru.getResourcePriority();
    if (ru.getAppID().getOID() == PRIORITIZED_ORGANIZATION)
        return p;
    if (p >= ResourcePriority.PRIORITY_UNKNOWN
        && p < ResourcePriority.PRIORITY_MSO_FIRST)
        return p;
    return ResourcePriority.PRIORITY_UNKNOWN;
}

public void resourceContentionWarning(ResourceUsage newRequest,
                                       ResourceUsage[] currentReservations)
{
    // Ignored for this example
}
**org.ocap.resource**

**Interface ApplicationResourceUsage**

**All Superinterfaces:**

ResourceUsage

```java
public interface ApplicationResourceUsage extends ResourceUsage
```

This interface represents a ResourceUsage corresponding to a resource explicitly reserved by an application by 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

<table>
<thead>
<tr>
<th>Methods inherited from interface org.ocap.resource.ResourceUsage</th>
</tr>
</thead>
<tbody>
<tr>
<td>getAppID, getResource, getResourceNames</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResourceUsage[] resolveResourceContention(ResourceUsage newRequest, ResourceUsage[] currentReservations)</td>
<td>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</td>
</tr>
<tr>
<td>void resourceContentionWarning(ResourceUsage newRequest, ResourceUsage[] currentReservations)</td>
<td>Warns the resource contention handler of an impending contention with a presenting ServiceContext (e.g., scheduled recording as defined by the OCAP DVR specification).</td>
</tr>
</tbody>
</table>

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 request(s).
- currentReservations - The set of resource usage objects that describe current resource reservations 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**

```java
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**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected ResourceContentionManager()</td>
<td>A constructor of this class.</td>
</tr>
</tbody>
</table>

---

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static ResourceContentionManager getInstance()</td>
<td>This method returns an instance of the ResourceContentionManager class.</td>
</tr>
<tr>
<td>int getWarningPeriod()</td>
<td>Gets the warning period set by the setWarningPeriod method.</td>
</tr>
<tr>
<td>void setResourceContentionHandler</td>
<td>This method sets the specified ResourceContentionHandler that decides which application shall be denied reserving a scarce resource.</td>
</tr>
<tr>
<td>(ResourceContentionHandler handler)</td>
<td></td>
</tr>
<tr>
<td>void setResourceFilter</td>
<td>This method sets an instance of a concrete class that extends AppsDatabaseFilter.</td>
</tr>
<tr>
<td>(AppsDatabaseFilter filter, java.lang.String resourceProxy)</td>
<td></td>
</tr>
<tr>
<td>void setWarningPeriod(int warningPeriod)</td>
<td>Sets the warning period used by the implementation to determine when to call the resourceContentionWarning method in a registered ResourceContentionHandler.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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(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)

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").
Interface ResourceUsage

All Known Subinterfaces:

- ApplicationResourceUsage
- RecordingResourceUsage
- ServiceContextResourceUsage
- SharedResourceUsage
- SharedResourceUsage
- TimeShiftBufferResourceUsage

public interface ResourceUsage

This interface represents a grouping of resources specific to an function performed by an application.

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AppID</strong></td>
</tr>
<tr>
<td>-</td>
</tr>
<tr>
<td><strong>ResourceProxy</strong></td>
</tr>
<tr>
<td>-</td>
</tr>
<tr>
<td><strong>java.lang.String[]</strong></td>
</tr>
<tr>
<td>-</td>
</tr>
</tbody>
</table>

Method Detail

gAppID

AppID  getAppID()

- Gets the AppID of the application associated with the set of resources contained in the usage.

Returns:

- AppID of the application.

gResourceNames

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.

gResource

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 provides information to distinguish which resources in the ResourceUsage have 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()
org.ocap.resource

Interface SharedResourceUsage

All Superinterfaces:
ResourceUsage

All Known Subinterfaces:
SharedResourceUsage

public interface SharedResourceUsage
extends ResourceUsage

This interface represents a group of resources where one or more resources are shared between multiple resource usages. If there is a contention over a resource that is considered shared between multiple uses, the shared usage SHALL be represented by an instance of SharedResourceUsage and the getResourceUsages() method would return the individual ResourceUsage instances that share the resource.

Because a SharedResourceUsage can contain multiple ResourceUsage instances where different entities reserved the resources, the value returned by the SharedResourceUsage.getAppID method SHALL be the AppID of the highest-priority ResourceUsage contained in the SharedResourceUsage or null if none of the contained ResourceUsages have AppIDs.

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getAppID()</td>
<td>Reports the AppID of the highest-priority ResourceUsage reported by getResourceUsages(), or null if none have AppIDs.</td>
</tr>
<tr>
<td>getResourceUsages()</td>
<td>Gets the list of ResourceUsage instances that share the resources represented by this resource usage.</td>
</tr>
<tr>
<td>getResourceUsages(ResourceProxy resource)</td>
<td>Gets the list of ResourceUsage instances that share a particular resource.</td>
</tr>
</tbody>
</table>

Methods inherited from interface org.ocap.resource.ResourceUsage

getResource, getResourceNames

Method Detail

getResourceUsages

getResourceUsages() | Gets the list of ResourceUsage instances that share the resources represented by this resource usage. |
| Returns: |
| An array of ResourceUsage instances that share one or more resources. |

getResourceUsages

getValue() | Gets the list of ResourceUsage instances that share a particular resource. |
| Parameters: |
resource - The shared resource for which ResourceUsage instances should be returned.

Returns:
An array of ResourceUsage instances that share the specified resource.

getAppID

AppID  getAppID()
Reports the AppID of the highest-priority ResourceUsage reported by getResourceUsages(), or null if none have AppIDs.

Specified by:
getAppID in interface ResourceUsage

Returns:
The AppID of the highest-priority ResourceUsage included in this SharedResourceUsage.
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 1.0.2] and [DVB-MHP 1.0.3]. The OpenCable Application Platform is in complete compliance with the following classes and interfaces from the org.dvb.media API:

<table>
<thead>
<tr>
<th>Table M–1 - DVB-GEM Specification Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.dvb.media Class</td>
</tr>
<tr>
<td>ActiveFormatDescriptionChangedEvent</td>
</tr>
<tr>
<td>AspectRatioChangedEvent</td>
</tr>
<tr>
<td>BackgroundVideoPresentationControl</td>
</tr>
<tr>
<td>CAException</td>
</tr>
<tr>
<td>CASStopEvent</td>
</tr>
<tr>
<td>DFCChangedEvent</td>
</tr>
<tr>
<td>DripFeedDataSource</td>
</tr>
<tr>
<td>DripFeedPermission</td>
</tr>
<tr>
<td>NoComponentSelectedEvent</td>
</tr>
<tr>
<td>PresentationChangedEvent</td>
</tr>
<tr>
<td>ServiceRemovedEvent</td>
</tr>
<tr>
<td>StopByResourceLossEvent</td>
</tr>
<tr>
<td>VideoFormatEvent</td>
</tr>
<tr>
<td>VideoFormatListener</td>
</tr>
</tbody>
</table>

M.2  Deviations from the DVB-MHP Specification

The information in this section deviates from the corresponding section in the [DVB-MHP 1.0.3]. The OpenCable Application Platform deviates from the following classes and interfaces in the org.dvb.media API:

| Table M–2 - Correlation between OCAP and [DVB-GEM 1.0.2] and DVB-MHP 1.0 |
|------------------------------------------------|----------------|----------------|----------------|----------------|
| OCAP                                          | [DVB-GEM 1.0.2]| GEM            | [DVB-MHP 1.0.3] | MHP            |
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 letterbox 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 1.0.2], 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-MHP 1.0.3] except for the following extension:

References to ETR 154 should be replaced with references to [SCTE 54].

*M.2.2.1.2 getInputVideoSize()*

```java
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-MHP 1.0.3] 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-MHP 1.0.3] 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-MHP 1.0.3] 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-MHP 1.0.3] except for the following extension:
References to ETR 154 should be replaced with references to [SCTE 54].

**M.3 OCAP Specific Requirements**

This information extends the specification requirements made to the [DVB-MHP 1.0.3].

An OCAP implementation SHALL be capable of presenting DTVCC contents for MPEG programs.
Annex N  Application Listing and Launching

This section 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-MHP 1.0.3] and [DVB-GEM 1.0.2]. The OpenCable Application Platform is in complete compliance with the API, org.dvb.application, specified in this section.

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>MHP Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annex N Application Listing and Launching</td>
<td>Annex S</td>
<td>Extension</td>
<td>Annex S</td>
<td>Extension</td>
</tr>
</tbody>
</table>

N.2  Deviations from the DVB-MHP Specification

There are no deviations from the corresponding section in the [DVB-GEM 1.0.2].

N.3  OCAP Specific Requirements

There are no additional OCAP requirements to [DVB-MHP 1.0.3].
Annex O  OCAP Fundamental Classes API

This section provides OCAP fundamental functions.

O.1  DVB-GEM and DVB-MHP Specification Correspondence

Annex O (this section) of OCAP corresponds to Annex I of [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2] as follows:

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>MHP Compliance</th>
</tr>
</thead>
</table>

O.2  OCAP Specific Requirements

There are no deviations from Annex I of [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2].

O.2.1  Extensions to DVB-MHP (Normative)

This information extends the specification requirements made to the [DVB-MHP 1.0.3] and [DVB-GEM 1.0.2].

The following information is normative for OCAP. It is specific to OCAP and is not contained within the [DVB-MHP 1.0.3] or [DVB-GEM 1.0.2].

O.2.1.1  org.ocap package

The org.ocap package provides the following fundamental OCAP classes:

**Package org.ocap**

Provides system utility functions.

**Class Summary**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OcapSystem</td>
<td>This class provides system utility functions.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void monitorConfiguredSignal()</td>
<td>Called by the Initial Monitor Application to inform the OCAP implementation it has completed its configuration process and that the boot processing may resume.</td>
</tr>
<tr>
<td>static void monitorConfiguringSignal(int port, int timeout)</td>
<td>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 20.2.2.3 Boot Process while connected to the cable network – CableCARD device present.</td>
</tr>
</tbody>
</table>

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, see Section 20.2.2.3 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 Service API

Table P–1 - Correlation between OCAP and [DVB-GEM 1.0.2]

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annex P OCAP Service API</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
</tbody>
</table>

Package org.ocap.service

The org.ocap.service package represents various aspects of a service.

See:

Description

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. |
| AlternativeContentErrorEvent | AlternativeContentErrorEvent is generated to indicate that "alternative" content is being presented due to an error that prevents the presentation of normal content as part of selection of a service and during presentation of that selected service. |
| S3DAlternativeContentErrorEvent | S3DAlternativeContentErrorEvent is generated to indicate that "alternative" content may be presenting due to a detected incompatibility between the selected 3D content and the display device. |
| ServiceTypePermission | ServiceTypePermission represents application permission to select a specific service type using a ServiceContext accessible by the application. |

Package org.ocap.service Description

The org.ocap.service package represents various aspects of a service.
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.util.Enumeration getAppAttributes()</td>
<td>Returns a java.util.Enumeration of references to org.ocap.application.OcapAppAttributes instances.</td>
</tr>
<tr>
<td>java.util.Enumeration getAppIDs()</td>
<td>Returns a java.util.Enumeration of references to org.dvb.application.AppID instances.</td>
</tr>
</tbody>
</table>

### Method Detail

#### getAppIDs

```java
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
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

<table>
<thead>
<tr>
<th>static javax.tv.service.ServiceType</th>
<th>OCAP_ABSTRACT_SERVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceType for an abstract service.</td>
<td></td>
</tr>
</tbody>
</table>

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)

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

Class AlternativeContentErrorEvent

java.lang.Object
  java.util.EventObject
    javax.tv.service.selection.ServiceContextEvent
      javax.tv.service.selection.PresentationChangedEvent
        javax.tv.service.selection.AlternativeContentEvent
          org.ocap.service.AlternativeContentErrorEvent

All Implemented Interfaces:
  java.io.Serializable

Direct Known Subclasses:
  S3DAlternativeContentErrorEvent

public class AlternativeContentErrorEvent
  extends javax.tv.service.selection.AlternativeContentEvent

AlternativeContentErrorEvent is generated to indicate that "alternative" content is being presented due to an error that prevents the presentation of normal content as part of selection of a service and during presentation of that selected service.

This event will be generated instead of SelectionFailedEvent where normal content could not be presented due to the following situations:

- The parental control settings prevent it.
- The CA system refusing to permit it.
- The requested content could not be found in the network.
- The absence of a ServiceContentHandler required to present the requested service.
- Problems with tuning.

Such presentation failures are not considered selection failures.

This event will be generated instead of PresentationTerminatedEvent where normal content presentation could not continue due to the following situations:

- The parental control settings prevent it.
- The CA system refusing to permit it.
- Inability to locate the requested content on the network.
- The absence of a ServiceContentHandler required to present the requested service.
- Change of tuning information.

Such presentation failures do not terminate presentation and allow for restoration of normal content presentation after correction of the error condition.

Note: The set of reason codes defined in this class may be extended by subclasses. Care should be taken to ensure that the values of newly-defined reason codes are unique.
See Also:

See Also: SelectionFailedEvent, PresentationTerminatedEvent, AlternativeContentEvent, Serialized Form

Field Summary

| static int | CA_REFUSAL | Reason code: Normal content could not be presented due to the CA system refusing to permit it. |
| static int | CONTENT_NOT_FOUND | Reason code: Normal content could not be presented because the requested content could not be found in the network. |
| static int | MISSING_HANDLER | Reason code : Normal content could not be presented due to absence of a ServiceContentHandler required to present the requested service's content. |
| static int | RATING_PROBLEM | Reason code: Normal content could not be presented due to a parental control rating problem. |
| static int | TUNING_FAILURE | Reason code : Normal content could not be presented due to problems with tuning. |

Fields inherited from class java.util.EventObject

source

Constructor Summary

AlternativeContentErrorEvent(javax.tv.service.selection.ServiceContext source, int reason)

Constructs an event with a reason code.

Method Summary

| int | getReason() | Reports the reason why alternative content is being presented. |

Methods inherited from class javax.tv.service.selection.ServiceContextEvent

getServiceContext

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

RATING_PROBLEM
public static final int RATING_PROBLEM
   Reason code: Normal content could not be presented due to a parental control rating problem.
   See Also:
   Constant Field Values

CA_REFUSAL
public static final int CA_REFUSAL
   Reason code: Normal content could not be presented due to the CA system refusing to permit it.
   See Also:
   Constant Field Values

CONTENT_NOT_FOUND
public static final int CONTENT_NOT_FOUND
   Reason code: Normal content could not be presented because the requested content could not be found in
   the network.
   See Also:
   Constant Field Values

MISSING_HANDLER
public static final int MISSING_HANDLER
   Reason code: Normal content could not be presented due to absence of a ServiceContentHandler
   required to present the requested service's content.
   See Also:
   ServiceContentHandler, Constant Field Values

TUNING_FAILURE
public static final int TUNING_FAILURE
   Reason code: Normal content could not be presented due to problems with tuning. This includes lack of
   tuning information as well as errors encountered during tuning.
   See Also:
   Constant Field Values

Constructor Detail

AlternativeContentErrorEvent
public AlternativeContentErrorEvent(javax.tv.service.selection.ServiceContext source,
   int reason)

   Constructs an event with a reason code.
   Parameters:
   source - The ServiceContext that generated the event.
   reason - The reason why alternative content is being presented.
Method Detail

getReason

public int getReason()

    Reports the reason why alternative content is being presented.
    Returns:
    The reason why alternative content is being presented. This SHALL be one of RATING_PROBLEM,
    CA_REFUSAL, CONTENT_NOT_FOUND, MISSING_HANDLER, or TUNING_FAILURE.
org.ocap.service

**Interface ServiceContextResourceUsage**

All Superinterfaces:
- ResourceUsage

```java
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getRequestedService()</code></td>
<td>Gets the Service that was requested when the resource contention was incurred.</td>
</tr>
<tr>
<td><code>getServiceContext()</code></td>
<td>Gets the ServiceContext for which the resources have been reserved.</td>
</tr>
</tbody>
</table>

Methods inherited from interface org.ocap.resource.ResourceUsage
- `getAppID`, `getResource`, `getResourceNames`

### Method Detail

**getRequestedService**

```java
javax.tv.service.Service getRequestedService()
```

Gets the Service that was requested when the resource contention was incurred.

**getServiceContext**

```java
javax.tv.service.selection.ServiceContext getServiceContext()
```

Gets the ServiceContext for which the resources have been reserved.

**getRequestedService**

```java
javax.tv.service.Service getRequestedService()
```

Gets the Service that was requested when the resource contention was incurred.

**Returns:**
- Service requested.
Class S3D Alternative Content Error Event

S3D Alternative Content Error Event is generated to indicate that "alternative" content may be presenting due to a detected incompatibility between the selected 3D content and the display device. The configuration of the Host device determines the nature of the alternative content presented; e.g., no video or the 3D video as requested. If the Host device attempts to present the requested 3D video, it is possible that it is being incorrectly displayed on the display device.

A device detects 3D content metadata (i.e., frame packing) based on signaling as defined in [OCCEP]. The implementation SHALL compare the signaled content format to HDMI display device capabilities reported in E-EDID and generate this event as warranted. This event will be generated due to the following situations:

- 3D video content selected but 3D video not supported by the HDMI display device.
- 3D video content selected but 3D format not supported by the HDMI display device.
- 3D video content selected but no display device connected to an HDMI port.

Such events are not considered selection or presentation failures.

Note (informative):

- If 3D video is selected for presentation but the Host device detects that no display devices are connected to an HDMI port, it will black out the video and mute the audio on any other connected outputs. See [HOST 2.1]
- If the Host device detects a possible incompatibility between the 3D content format and the display device, it may black out the video or send the requested 3D video to the display device, depending on the Host device configuration per [HOST 2.1] and [MIB-HOST].

See Also:

- Serialized Form

Field Summary

<table>
<thead>
<tr>
<th>static int S3D_FORMAT_NOT_SUPPORTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason code: 3D video content has been selected but its 3D format is not supported by the HDMI display device.</td>
</tr>
</tbody>
</table>
## Field Summary

<table>
<thead>
<tr>
<th>static int</th>
<th>S3D_NO_HDMI_CONNECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reason code: 3D video content has been selected but no display is connected to an HDMI port.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>S3D_NOT_SUPPORTED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reason code: 3D video content has been selected but 3D video is not supported by the HDMI display device.</td>
</tr>
</tbody>
</table>

### Fields inherited from class org.ocap.service.AlternativeContentErrorEvent

- CA_REFUSAL, CONTENT_NOT_FOUND, MISSING_HANDLER, RATING_PROBLEM, TUNING_FAILURE

### Fields inherited from class java.util.EventObject

- source

### Constructor Summary

S3DAlternativeContentErrorEvent(javax.tv.service.selection.ServiceContext source, int reason)

Constructs an event with a reason code.

### Method Summary

#### Methods inherited from class org.ocap.service.AlternativeContentErrorEvent

- getReason

#### Methods inherited from class javax.tv.service.selection.ServiceContextEvent

- getServiceContext

#### 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

### Field Detail

**S3D_NOT_SUPPORTED**

public static final int S3D_NOT_SUPPORTED

Reason code: 3D video content has been selected but 3D video is not supported by the HDMI display device.

See Also:

Constant Field Values
S3D_FORMAT_NOT_SUPPORTED
public static final int S3D_FORMAT_NOT_SUPPORTED
    Reason code: 3D video content has been selected but its 3D format is not supported by the HDMI display device.
    See Also:
    Constant Field Values

S3D_NO_HDMI_CONNECTION
public static final int S3D_NO_HDMI_CONNECTION
    Reason code: 3D video content has been selected but no display is connected to an HDMI port.
    See Also:
    Constant Field Values

Constructor Detail
S3DAlternativeContentErrorEvent
public S3DAlternativeContentErrorEvent(javax.tv.service.selection.ServiceContext source,
                                            int reason)
    Constructs an event with a reason code.
    Parameters:
    source - The ServiceContext that generated the event.
    reason - The reason why alternative content is potentially being presented.
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.

See Also:
    Serialized Form

Field Summary

<table>
<thead>
<tr>
<th>static java.lang.String</th>
<th>BROADCAST</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Indicates an inband broadcast service provided by a content provider.</td>
</tr>
<tr>
<td>static java.lang.String</td>
<td>MFR</td>
</tr>
<tr>
<td></td>
<td>Indicates an abstract service provided by the Host device manufacturer.</td>
</tr>
<tr>
<td>static java.lang.String</td>
<td>MSO</td>
</tr>
<tr>
<td></td>
<td>Indicates an abstract service provided by the HFC network provider (i.e., MSO).</td>
</tr>
</tbody>
</table>

Constructor Summary

ServiceTypePermission(java.lang.String type, java.lang.String actions)
    Creates a new ServiceTypePermission object with the specified service type name.

Method Summary

<table>
<thead>
<tr>
<th>boolean equals(java.lang.Object obj)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tests two ServiceTypePermission objects for equality.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>java.lang.String getActions()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the canonical representation of the actions string.</td>
</tr>
</tbody>
</table>
Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int hashCode</td>
<td></td>
<td>Provides the hash code value of this object.</td>
</tr>
<tr>
<td>boolean implies</td>
<td>(java.security.Permission p)</td>
<td>Checks if the specified permission is &quot;implied&quot; by this object.</td>
</tr>
</tbody>
</table>

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.

See Also:
Constant Field Values

MSO

public static final java.lang.String MSO

Indicates an abstract service provided by the HFC network provider (i.e., MSO).

See Also:
Constant Field Values

BROADCAST

public static final java.lang.String BROADCAST

Indicates an inband broadcast service provided by a content provider.

See Also:
Constant Field Values

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 [26] ISO/IEC 13818-1 defines broadcast source_id values that correspond to a broadcast type string and table 11-4 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

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

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 System API

This section presents the \texttt{org.ocap.system} APIs. This API is used to access system modules described in Section 20.2.2.

\textbf{Table Q–1 - Correlation between OCAP and [DVB-GEM 1.0.2]}

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annex Q OCAP System API</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
</tbody>
</table>

\textbf{Package org.ocap.system}

This API is used to access system modules.

See:

\textbf{Description}

\textbf{Interface Summary}

- \textbf{EASHandler} An OCAP-J application can register an EASHandler to the EASModuleRegistrar via the \texttt{EASModuleRegistrar.registerEASHandler} (\texttt{org.ocap.system.EASHandler}) method.

- \textbf{EASListener} This interface represents a listener that will receive EAS events.

- \textbf{SystemModule} The SystemModule is used by an OCAP-J application to send an APDU to the CableCARD device.

- \textbf{SystemModuleHandler} The SystemModuleHandler is used by an OCAP-J application 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.

\textbf{Class Summary}

- \textbf{EASEvent} This class represents an EAS event.

- \textbf{EASManager} This class represents a manager that allows applications to register listeners for EAS events.

- \textbf{EASModuleRegistrar} An OCAP-J application can set and get a preferred attribute value of an EAS alert text.

- \textbf{MonitorAppPermission} The MonitorAppPermission class represents permission to execute privileged operations only Monitor Application should be granted.

- \textbf{RegisteredApiManager} This class represents a manager for registered APIs that can be registered with an implementation by a privileged application.

- \textbf{RegisteredApiUserPermission} The RegisteredApiUserPermission class represents permission for an application to use a specific registered API.

- \textbf{SystemModuleRegistrar} This class is used by an OCAP-J application to access a system module.
Package org.ocap.system Description

This API is used to access system modules.

Example of application access of Specific Application Resource

This is sample code for accessing the Specific Application Resource. See also Section 20.2.2.4.1.

```java
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 a 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 accessing MMI Resource

This is sample code for accessing MMI Resource. See also Section 20.2.2.4.2.

```java
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 to access 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 registration.
     */
    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 of setting EAS Audio and Display Attributes

This is sample code for OCAP-J application management of audio presentation and setting of EASdisplay attributes. See also Section 20.2.3.10.

```java
import org.ocap.system.*;
```
/**
 * This is a sample implementation of EAS manager.
 */
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.
    }
}
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.

See Also:
   Serialized Form

Field Summary

<table>
<thead>
<tr>
<th>static int</th>
<th>EAS_COMPLETE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The event is generated when an EAS message completes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>EAS_DETAILS_CHANNEL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This event is generated when an EAS table is detected by the implementation and the table requires a forced tune.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>EAS_TEXT_DISPLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This event is generated when an EAS table is detected by the implementation and the table requires textual display.</td>
</tr>
</tbody>
</table>

Fields inherited from class java.util.EventObject

source

Constructor Summary

EASEvent(java.lang.Object source, int reason)

Constructs a EASEvent with the specified source and reason.

Method Summary

int getReason()

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.
  - See Also:
  - Constant Field Values

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.
  - See Also:
  - Constant Field Values

EAS_COMPLETE
- public static final int EAS_COMPLETE
  - The event is generated when an EAS message completes.
  - See Also:
  - Constant Field Values

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

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>notifyPrivateDescriptor(byte[] descriptor)</td>
<td>This is a call back method to notify a private descriptor in the cable_emergency_alert message defined in [SCTE 18].</td>
</tr>
<tr>
<td>stopAudio()</td>
<td>This is a call back method to notify that the alert duration has finished.</td>
</tr>
</tbody>
</table>

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]. If the alert_priority=15 but no audio specified by [SCTE 18] 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]. 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].

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>notify(EASEvent event)</td>
<td>Notifies an application that an EAS message has started, is in-progress, or has completed.</td>
</tr>
<tr>
<td>void</td>
<td>warn(EASEvent event)</td>
<td>Warns an application that an EAS message has been received and parsed and resources will soon be allocated to it.</td>
</tr>
</tbody>
</table>

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 |
|           | Indicates the implementation is processing the EAS message and EAS information is being presented. |
| static int | EAS_MESSAGE_RECEIVED_STATE   |
|           | Indicates an EAS message has been received and is about to be processed. |
| static int | EAS_NOT_IN_PROGRESS_STATE    |
|           | Indicates an EAS message is not being processed. |

Constructor Summary

EASManager()  

Method Summary

| abstract void | addListener(EASListener listener) |
|              | Adds a listener for EAS events. |
| static EASManager | getInstance() |
|                   | Gets the instance of the EAS Manager class that may be used by the application to register an EASListener. |
| int | getState() |
|      | Gets the EAS state. |
| abstract void | removeListener(EASListener listener) |
|                | 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

```java
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.

See Also:
Constant Field Values

EAS_MESSAGE_IN_PROGRESS_STATE

```java
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.

See Also:
Constant Field Values

EAS_NOT_IN_PROGRESS_STATE

```java
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.

See Also:
Constant Field Values

Constructor Detail

EASManager

```java
public EASManager()
```

Method Detail

getInstance

```java
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

```java
public abstract void addListener(EASListener listener)
```
Adds a listener for EAS events.

Parameters:
listener - The new EAS listener.

removeListener

```java
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**

```java
public int getState()
```

Gets the EAS state. Possible return values are defined by state constants in this class.

**Returns:**
EAS state.
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 of a private descriptor indicating an alternative audio, if the alert_priority=15 but no audio specified by SCTE 18 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 detail on EAS functionality.

Since:
1.0

### Field Summary

<table>
<thead>
<tr>
<th>static int</th>
<th>EAS_ATTRIBUTE_BACK_COLOR</th>
<th>Indicates a background color attribute of an EAS alert text.</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int</td>
<td>EAS_ATTRIBUTE_BACK_OPACITY</td>
<td>Indicates a background opacity attribute of an EAS alert text.</td>
</tr>
<tr>
<td>static int</td>
<td>EAS_ATTRIBUTE_FONT_COLOR</td>
<td>Indicates a font color attribute of an EAS alert text.</td>
</tr>
<tr>
<td>static int</td>
<td>EAS_ATTRIBUTE_FONT_FACE</td>
<td>Indicates a font face attribute of an EAS alert text.</td>
</tr>
<tr>
<td>static int</td>
<td>EAS_ATTRIBUTE_FONT_OPACITY</td>
<td>Indicates a font opacity attribute of an EAS alert text.</td>
</tr>
<tr>
<td>static int</td>
<td>EAS_ATTRIBUTE_FONT_SIZE</td>
<td>Indicates a font size attribute of an EAS alert text.</td>
</tr>
<tr>
<td>static int</td>
<td>EAS_ATTRIBUTE_FONT_STYLE</td>
<td>Indicates a font style attribute of an EAS alert text.</td>
</tr>
</tbody>
</table>

### Constructor Summary

| protected    | EASModuleRegistrar()    | A constructor of this class.                               |
Method Summary

<table>
<thead>
<tr>
<th>Java Method</th>
<th>Method Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>java.lang.Object getEASAttribute(int attribute)</code></td>
<td>This method returns a current attribute values applied to an EAS alert text on a screen.</td>
</tr>
<tr>
<td><code>java.lang.Object[] getEASCapability(int attribute)</code></td>
<td>This method returns a possible attribute values applied to an EAS alert text on a screen.</td>
</tr>
<tr>
<td><code>static EASModuleRegistrar getInstance()</code></td>
<td>This method returns a sole instance of the EASModuleRegistrar class.</td>
</tr>
<tr>
<td><code>void registerEASHandler(EASHandler handler)</code></td>
<td>This method registers an EASHandler instance.</td>
</tr>
<tr>
<td><code>void setEASAttribute(int[] attribute, java.lang.Object[] value)</code></td>
<td>This method sets a preferred attribute values applied to an EAS alert text on a screen.</td>
</tr>
<tr>
<td><code>void unregisterEASHandler()</code></td>
<td>This method unregisters the current registered EASHandler instance.</td>
</tr>
</tbody>
</table>

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.

See Also:

Constant Field Values

**EAS_ATTRIBUTE_FONT_STYLE**

public static final int EAS_ATTRIBUTE_FONT_STYLE

Indicates a font style attribute of an EAS alert text.

See Also:

Constant Field Values

**EAS_ATTRIBUTE_FONT_FACE**

public static final int EAS_ATTRIBUTE_FONT_FACE

Indicates a font face attribute of an EAS alert text.

See Also:

Constant Field Values

**EAS_ATTRIBUTE_FONT_SIZE**

public static final int EAS_ATTRIBUTE_FONT_SIZE

Indicates a font size attribute of an EAS alert text.
See Also:
Constant Field Values

EAS_ATTRIBUTE_BACK_COLOR
public static final int EAS_ATTRIBUTE_BACK_COLOR
Indicates a background color attribute of an EAS alert text.
See Also:
Constant Field Values

EAS_ATTRIBUTE_FONT_OPACITY
public static final int EAS_ATTRIBUTE_FONT_OPACITY
Indicates a font opacity attribute of an EAS alert text.
See Also:
Constant Field Values

EAS_ATTRIBUTE_BACK_OPACITY
public static final int EAS_ATTRIBUTE_BACK_OPACITY
Indicates a background opacity attribute of an EAS alert text.
See Also:
Constant Field Values

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_ATTRIBUTE_FONT_OPACITY or
EAS_ATTRIBUTE_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.

<table>
<thead>
<tr>
<th>Permission Name</th>
<th>What the Permission Allows</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>registrar</td>
<td>Provides access to the Application Database by way of the AppRegistrar</td>
<td>This permission allows the caller to add or remove applications from the Application Database.</td>
</tr>
<tr>
<td>service</td>
<td>Allows creation of an AbstractService</td>
<td>Applications with this permission can create and manage their own service contexts and the services running in those service contexts.</td>
</tr>
<tr>
<td>servicemanager</td>
<td>Allows management of all services</td>
<td>Applications with this permission can create their own service contexts and manage both their own and other service contexts and services.</td>
</tr>
<tr>
<td>security</td>
<td>Allows setting the SecurityPolicyHandler used by the AppManager</td>
<td>This permission allows the application to register a SecurityPolicyHandler with the AppManager to determine the PermissionCollection granted to applications before they are run.</td>
</tr>
<tr>
<td>reboot</td>
<td>Initiates a system to reboot itself</td>
<td>This permission allows the caller to request for a system reboot.</td>
</tr>
<tr>
<td>systemevent</td>
<td>Allows setting the error, resource depletion, or reboot handlers</td>
<td>This permission allows the Monitor Application to be alerted upon system reboot, resource depletion, and error events.</td>
</tr>
<tr>
<td>Permission Name</td>
<td>What the Permission Allows</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>handler.appFilter</td>
<td>Set a new black and white list to the system</td>
<td>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.</td>
</tr>
<tr>
<td>handler.resource</td>
<td>Set a Resource Contention Handler</td>
<td>Set a handler to resolve resource contention between two or more apps see ResourceContentionManager.setResourceContentionHandler (org.ocap.resource. ResourceContentionHandler).</td>
</tr>
<tr>
<td>handler.closedCaptioning</td>
<td>Set closed-captioning preferences and control captioning.</td>
<td>Allows monitor application to get a ClosedCaptioningAttribute and call methods in a ClosedCaptioningControl.</td>
</tr>
<tr>
<td>filterUserEvents</td>
<td>Filter user events</td>
<td>This permission allows the user to filter user events.</td>
</tr>
<tr>
<td>handler.eas</td>
<td>Set preferences of Emergency Alert System (EAS) message representation.</td>
<td>Allows monitor application to set preferences of EAS message representation and add a new EASHandler by calling EASModuleRegistrar .registerEASHandler (org.ocap.system.EASHandler).</td>
</tr>
<tr>
<td>setVideoPort</td>
<td>Allows enabling and disabling video port</td>
<td>Allows monitor to call org.ocap.hardware.VideoOutputPort.enable() and org.ocap.hardware.VideoOutputPort.disable().</td>
</tr>
<tr>
<td>podApplication</td>
<td>Allows access to Specific Application Support Resource</td>
<td>Allows Monitor Application to call org.ocap.system.SystemModuleRegistrar.</td>
</tr>
<tr>
<td>signal.configured</td>
<td>Allows monitor to signal implementation to resume boot processing after handlers have been set</td>
<td>Allows monitor to call org.ocap.OcapSystem.monitorConfiguredSignal().</td>
</tr>
<tr>
<td>storage</td>
<td>Provides control of persistent storage devices and content stored therein</td>
<td>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.</td>
</tr>
<tr>
<td>properties</td>
<td>Allows monitor to access ocap system properties</td>
<td>Allows monitor to call read ocap properties that require monitor application permission.</td>
</tr>
<tr>
<td>registeredapi.manager</td>
<td>Provides access to network specific APIs</td>
<td>Gives monitor ability to register an API, remove a registered API, or access a registered API.</td>
</tr>
<tr>
<td>vbifiltering</td>
<td>Allows monitor application to filter VBI data.</td>
<td>Allows monitor application to call a VBIFilterGroup constructor.</td>
</tr>
<tr>
<td>Permission Name</td>
<td>What the Permission Allows</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>codeDownload</td>
<td>Allows monitor application to initiate a download, generally following a CVT signaling a deferred download</td>
<td>Allow monitor application to call Host.codeDownload method.</td>
</tr>
<tr>
<td>mediaAccess</td>
<td>Allows monitor application to register MediaAccessHandler.</td>
<td>Allows monitor application to call MediaAccessHandlerRegistrar.setExternalTriggers(). Allows monitor application to call a MediaAccessConditionControl.conditionHasChanged().</td>
</tr>
<tr>
<td>powerMode</td>
<td>Allows an application to set the power mode.</td>
<td>Applications with this permission can programmatically control the power mode of the device.</td>
</tr>
<tr>
<td>environment.selection</td>
<td>Allows monitor application to request the cable environment to become selected or deselected</td>
<td>Allows monitor application to request the cable environment to become selected or deselected by calling Environment.select or Environment.deselect</td>
</tr>
<tr>
<td>logger.config</td>
<td>Allows an application to configure the logger.</td>
<td>Applications with this permission can programmatically control the configuration of the logger using the log4j API.</td>
</tr>
</tbody>
</table>

Other permissions may be added as necessary.

See Also:
- Serialized Form

## 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

<table>
<thead>
<tr>
<th>protected RegisteredApiManager()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protected constructor.</td>
</tr>
</tbody>
</table>

Method Summary

| static RegisteredApiManager getInstance() | Gets the singleton instance of the Registered API manager. |
|------------------------------------------|
| abstract java.lang.String[] getNames() | Gets a list of registered APIs. |
| abstract java.lang.String[] getUsedNames() | Gets a list of registered APIs that are in use by the caller. |
| abstract java.lang.String getVersion(java.lang.String name) | Gets the version of a registered API, or null if it is not registered. |
| abstract void unregister(java.lang.String name) | 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


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.IllegalArgumentException - 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)

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.

See Also:
   Serialized Form

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegisteredApiUserPermission(java.lang.String name)</td>
<td>Creates a new RegisteredApiUserPermission with the specified name.</td>
</tr>
<tr>
<td>RegisteredApiUserPermission(java.lang.String name, java.lang.String actions)</td>
<td>Creates a new RegisteredApiUserPermission with the specified name.</td>
</tr>
</tbody>
</table>

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**

```java
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.
public interface SystemModule

The SystemModule is used by an OCAP-J application to send an APDU to the CableCARD device. A SystemModule instance is provided by the SystemModuleHandler.ready(SystemModule) method after calling the SystemModuleRegistrar.registerSASHandler(SystemModuleHandler, byte[]) method or the SystemModuleRegistrar.registerMMIHandler(SystemModuleHandler) method.

See Also: SystemModuleRegistrar

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void sendAPDU</td>
<td>This method sends an APDU to the CableCARD device.</td>
</tr>
</tbody>
</table>

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 [CCIF 2.0] 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 registered via the SystemModuleRegistrar.registerSASHandler(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 Resource and Application Information Resource is 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 InterfaceSpecification.

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.
**Interface SystemModuleHandler**

```
public interface SystemModuleHandler
```

The SystemModuleHandler is used by an OCAP-J application 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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void notifyUnregister()</td>
<td>This is a call back method to notify that the SystemModuleHandler is being unregistered and give a chance to do a termination procedure.</td>
</tr>
<tr>
<td>void ready(SystemModule systemModule)</td>
<td>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.</td>
</tr>
<tr>
<td>void receiveAPDU(int apduTag, int lengthField, byte[] dataByte)</td>
<td>This is a call back method to notify an APDU received from the CableCARD device.</td>
</tr>
<tr>
<td>void sendAPDUFailed(int apduTag, byte[] dataByte)</td>
<td>This is a call back method to notify an error has occurred while sending an APDU via the SystemModule.sendAPDU(int, byte[]) method.</td>
</tr>
</tbody>
</table>

### 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 [CCIF 2.0] 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 receives 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 [CCIF 2.0].

**sendAPDUFailed**

```java
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**

```java
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**

```java
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 access a system module.

Private Host Application
An OCAP-J application MAY register a SystemModuleHandler to act as 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
An OCAP-J application MAY register a SystemModuleHandler to access the Host’s MMI and Application
Information resources. If the SystemModuleHandler is registered successfully, the Host’s MMI Resource and
Application Information Resource MAY not terminate, but the implementation SHALL pass all APDUs to the
registered Handler, and the resident MMI dialog SHALL be hidden.

See also Section 20 Baseline Functionality for details.

Since: 1.0

Constructor Summary

protected SystemModuleRegistrar()
A constructor of this class.

Method Summary

static SystemModuleRegistrar getInstance()
This method returns a sole instance of the SystemModuleRegistrar
class.

void registerMMIHandler(SystemModuleHandler handler)
This method registers a SystemModuleHandler instance for
accessing MMI and Application Information Resource.

void registerSASHandler(SystemModuleHandler handler,
byte[] privateHostAppID)
This method registers the specified SystemModuleHandler
instance for the specified privateHostAppID.

void unregisterMMIHandler()
This method unregisters the SystemModuleHandler and
SystemModule instance of the registered application accessing MMI and
Application Information Resource and revives the resident MMI and
Application Information Resource.
Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void unregisterSASHandler(byte[] privateHostAppID)</td>
<td>This method unregisters the SystemModuleHandler and the SystemModule instance corresponding to the specified privateHostAppID, and revives an original resident Private Host Application.</td>
</tr>
<tr>
<td>void unregisterSASHandler(SystemModuleHandler handler)</td>
<td>This method unregisters the specified SystemModuleHandler and the corresponding SystemModule instance, and revives an original resident Private Host Application.</td>
</tr>
</tbody>
</table>

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 registered 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.
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 accessing 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.IllegalArgumentException - if the second SystemModuleHandler is to be registered.

unregisterMMIHandler

public void unregisterMMIHandler()

This method unregisters the SystemModuleHandler and SystemModule instance of the registered application accessing 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 Hardware POD API

This section presents the org.ocap.hardware.pod APIs.

<table>
<thead>
<tr>
<th></th>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annex R OCAP Hardware POD API</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
<td></td>
</tr>
</tbody>
</table>

**Package org.ocap.hardware.pod**

This package provides a way to set and get OpenCable CableCARD Resource related parameters.

**Interface Summary**

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HostParamHandler</td>
<td>A class that implements this interface can reject the update of the Feature parameter in the Host device.</td>
</tr>
<tr>
<td>PODApplication</td>
<td>This class represents an Application that resides in the OpenCable CableCARD device.</td>
</tr>
</tbody>
</table>

**Class Summary**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POD</td>
<td>This class provides an access to functions and information of the OpenCable CableCARD device on the OCAP Host device.</td>
</tr>
</tbody>
</table>

**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**. 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:

```java
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(HostParamHandler) method.

Example:

```java
import org.ocap.hardware.pod.*;
...
POD pod = POD.getInstance();
int acOutlet = 7;
byte[] unswitched = {0X02};
byte[] value = pod.getHostParam(acOutlet);
...
pod.updateHostParam(acOutlet, unswitched);
...
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 the [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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>notifyUpdate(int featureID, byte[] value)</td>
<td>This is a call back method to notify an update of the Feature parameter in the Host device.</td>
</tr>
</tbody>
</table>

### 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

```java
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() |
| A constructor of this class. |

Method Summary

| PODApplication[] | getApplications() |
| This method returns the CableCARD device applications listed in the Application_info_cnf() APDU defined in the OpenCable CableCARD Interface specification. |

| int[] | getHostFeatureList() |
| This method returns a list of the Feature IDs supported by the Host device. |

| byte[] | getHostParam(int featureID) |
| This method returns the current Feature parameter value in the Host device for the specified Feature ID. |

| static POD | getInstance() |
| This method returns the sole instance of the POD class. |

| int | getManufacturerID() |
| This method returns a CableCARD device manufacturer ID. |

| int | getVersionNumber() |
| This method returns a CableCARD device version number. |
## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>isReady()</td>
<td>This method provides a current status of the CableCARD device.</td>
</tr>
<tr>
<td>void</td>
<td>setHostParamHandler(HostParamHandler handler)</td>
<td>This method sets an instance of a class that implements the HostParamHandler interface.</td>
</tr>
<tr>
<td>boolean</td>
<td>updateHostParam(int featureID, byte[] value)</td>
<td>This method updates the Feature parameter value in the Host device.</td>
</tr>
</tbody>
</table>

## 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 [CCIF 2.0].
- **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 [CCIF 2.0].

Throws:
- java.lang.IllegalArgumentException - 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.IllegalArgumentException - 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 [CCIF 2.0] 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 [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 [CCIF 2.0]. 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 [CCIF 2.0]. For example, if the featureID is 0x1, the value is

Rf_output_channel() { 
    Output_channel
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 [CCIF 2.0]. See also the HostParamHandler for more information.

Parameters:
featureID - a Feature ID defined in [CCIF 2.0]. 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 [CCIF 2.0].

Field Summary

<table>
<thead>
<tr>
<th>static int TYPE_CA</th>
<th>The Conditional Access application type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int TYPE_CP</td>
<td>The &quot;Copy Protection&quot; application type.</td>
</tr>
<tr>
<td>static int TYPE_DIAGNOSTIC</td>
<td>The &quot;Diagnostic&quot; application type.</td>
</tr>
<tr>
<td>static int TYPE_DVS167</td>
<td>The &quot;Network Interface - DVS/167&quot; application type.</td>
</tr>
<tr>
<td>static int TYPE_DVS178</td>
<td>The &quot;Network Interface - DVS/178&quot; application type.</td>
</tr>
<tr>
<td>static int TYPE_IP</td>
<td>The &quot;IP Service&quot; application type.</td>
</tr>
<tr>
<td>static int TYPE_UNDESIGNATED</td>
<td>The &quot;Undesignated&quot; application type.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>java.lang.String getName()</th>
<th>This method returns an application name of the CableCARD Application represented by this class.</th>
</tr>
</thead>
<tbody>
<tr>
<td>int getType()</td>
<td>This method returns an application type value of the CableCARD Application represented by this class.</td>
</tr>
<tr>
<td>java.lang.String getURL()</td>
<td>This method returns a URL of the CableCARD Application represented by this class.</td>
</tr>
<tr>
<td>int getVersionNumber()</td>
<td>This method returns an application version number of the CableCARD Application represented by this class.</td>
</tr>
</tbody>
</table>
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 OpenCable CableCARD Interface specification.

See Also:
Constant Field Values

**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 OpenCable CableCARD Interface specification.

See Also:
Constant Field Values

**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 OpenCable CableCARD Interface specification.

See Also:
Constant Field Values

**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].

See Also:
Constant Field Values

**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].

See Also:
Constant Field Values

**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].

See Also:
Constant Field Values

**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].

See Also:
Constant Field Values

Method Detail

**getType**

```java
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**

```java
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
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
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 Media API

This section presents the org.ocap.media APIs.

Table S–1 - Correlation between OCAP and [DVB-GEM 1.0.2]

<table>
<thead>
<tr>
<th></th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annex S OCAP Media API</td>
<td>No Corresponding Section</td>
<td>OCAP-Specific Extension</td>
</tr>
</tbody>
</table>

Package org.ocap.media

The org.ocap.media package is a collection of classes and interfaces for controlling access to various kinds of media components

See: Description

Interface Summary

<table>
<thead>
<tr>
<th>Class/Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlternativeMediaPresentationReason</td>
<td>This interface represents possible reasons that lead to alternative media presentation.</td>
</tr>
<tr>
<td>ClosedCaptioningControl</td>
<td>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.</td>
</tr>
<tr>
<td>ClosedCaptioningListener</td>
<td>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.</td>
</tr>
<tr>
<td>MediaAccessAuthorization</td>
<td>A MediaAccessAuthorization object represents the presentation authorization given by a registered MediaAccessHandler for a specific A/V content.</td>
</tr>
<tr>
<td>MediaAccessConditionControl</td>
<td>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.</td>
</tr>
<tr>
<td>MediaAccessHandler</td>
<td>A class implementing this interface can prevent the presentation of A/V service components.</td>
</tr>
<tr>
<td>MediaTimerListener</td>
<td>This is a listener to inform events on a MediaTimer object.</td>
</tr>
<tr>
<td>NotPresentedMediaInterface</td>
<td>NotPresentedMediaInterface shall be implemented by classes which can report failure to access media components.</td>
</tr>
<tr>
<td>S3DConfiguration</td>
<td>This interface represents a 3D frame packing payload as defined in [OCCEP].</td>
</tr>
<tr>
<td>S3DFormatTypes</td>
<td>The interface contains constants representing video formats for stereoscopic 3D streams.</td>
</tr>
<tr>
<td>VBIFilter</td>
<td>This class represents a VBI filter.</td>
</tr>
<tr>
<td>VBIFilterListener</td>
<td>This interface represents a VBI filter event listener.</td>
</tr>
</tbody>
</table>
Interface Summary

| VideoFormatControl | This interface extends \texttt{org.dvb.media.VideoFormatControl} to provide access to OCAP-specific info signaled in presented video, such as 3D formatting data. |

Class Summary

| AlternativeMediaPresentationEvent | \texttt{AlternativeMediaPresentationEvent} is a JMF event generated to indicate that an "alternative" content is presented during the media presentation of a service. |
| 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 \texttt{AlternativeMediaPresentationEvent} or a \texttt{NormalMediaPresentationEvent}. |
| MediaPresentationEvent | \texttt{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 | \texttt{NormalMediaPresentationEvent} is a JMF event generated when the normal media components of a service are presented. |
| S3DSignalingChangedEvent | This class represents an event that will be reported to an application with an \texttt{org.ocap.media.VideoFormatControl}. |
| VBIFilterEvent | This class represents a VBI filter event. |
| VBIFilterGroup | This class represents a group of VBI data filters. |

Package \texttt{org.ocap.media} Description

The \texttt{org.ocap.media} package is a collection of classes and interfaces for controlling access to various kinds of media components.
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.
See Also:
   Serialized Form

Field Summary

Fields inherited from class java.util.EventObject
source

Constructor Summary

protected AlternativeMediaPresentationEvent(javax.media.Controller from,
   int previous, int current, int target)
Constructor of MediaPresentationEvent

Method Summary

ElementaryStream[] getNotPresentedStreams()
### Method Summary

| int | getReason(ElementaryStream es) |

Methods inherited from class `org.ocap.media.MediaPresentationEvent`

- getPresentedStreams, getSourceURL, getTrigger, isSourceDigital

Methods inherited from class `javax.media.TransitionEvent`

- getCurrentState, getPreviousState, getTargetState

Methods inherited from class `javax.media.ControllerEvent`

- getSource, getSourceController

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

**AlternativeMediaPresentationEvent**

protected `AlternativeMediaPresentationEvent`(`javax.media.Controller` from, int previous, int current, int target)

Constructor of MediaPresentationEvent

See Also:

- MediaPresentationEvent

### Method Detail

**getNotPresentedStreams**

public `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(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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int BROADCAST_INCONSISTENCY</td>
<td>Bit indicating that broadcast information is inconsistent: for example PMT is missing.</td>
</tr>
<tr>
<td>static int CA_UNKNOWN</td>
<td>Bit indicating that media are ciphered and the CA does not correspond to ciphering.</td>
</tr>
<tr>
<td>static int COMMERCIAL_DIALOG</td>
<td>Bit indicating that a user dialog for payment is necessary before media presentation.</td>
</tr>
<tr>
<td>static int HARDWARE_RESOURCE_NOT_AVAILABLE</td>
<td>Bit indicating that hardware resource necessary for presenting service components is not available.</td>
</tr>
<tr>
<td>static int NO_ENTITLEMENT</td>
<td>Bit indicating that service components are ciphered and the user has no entitlement to view all or part of them.</td>
</tr>
<tr>
<td>static int RATING_PROBLEM</td>
<td>Reason indicating that media presentation is not authorized according to the program rating.</td>
</tr>
<tr>
<td>static int REASON_FIRST</td>
<td>Marks the first bit for the range of alternative media presentation reasons.</td>
</tr>
<tr>
<td>static int REASON_LAST</td>
<td>Marks the last bit for the range of alternative media presentation reasons.</td>
</tr>
</tbody>
</table>

Field Detail

REASON_FIRST

static final int REASON_FIRST
Marks the first bit for the range of alternative media presentation reasons.

See Also:
Constant Field Values

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.

See Also:
Constant Field Values
COMMERCIAL_DIALOG
static final int COMMERCIAL_DIALOG
    Bit indicating that a user dialog for payment is necessary before media presentation.
    See Also:
    Constant Field Values

RATING_PROBLEM
static final int RATING_PROBLEM
    Reason indicating that media presentation is not authorized regarding to the program rating.
    See Also:
    Constant Field Values

CA_UNKNOWN
static final int CA_UNKNOWN
    Bit indicating that media are ciphered and the CA does not correspond to ciphering.
    See Also:
    Constant Field Values

BROADCAST_INCONSISTENCY
static final int BROADCAST_INCONSISTENCY
    Bit indicating that broadcast information is inconsistent : for example PMT is missing.
    See Also:
    Constant Field Values

HARDWARE_RESOURCE_NOT_AVAILABLE
static final int HARDWARE_RESOURCE_NOT_AVAILABLE
    Bit indicating that hardware resource necessary for presenting service components is not available.
    See Also:
    Constant Field Values

REASON_LAST
static final int REASON_LAST
    Marks the last bit for the range of alternative media presentation reasons.
    See Also:
    Constant Field Values
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

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int CC_ATTRIBUTE_FONT_ITALICIZED</td>
</tr>
<tr>
<td>Indicates a font face attribute of a closed-captioning text.</td>
</tr>
<tr>
<td>static int CC_ATTRIBUTE_FONT_STYLE</td>
</tr>
<tr>
<td>Indicates a font style attribute of a closed-captioning text.</td>
</tr>
<tr>
<td>static int CC_ATTRIBUTE_FONT_UNDERLINE</td>
</tr>
<tr>
<td>Indicates a font face attribute of a closed-captioning text.</td>
</tr>
<tr>
<td>static int CC_ATTRIBUTE_PEN_BG_COLOR</td>
</tr>
<tr>
<td>Indicates a pen back ground color attribute to draw closed-captioning text.</td>
</tr>
<tr>
<td>static int CC_ATTRIBUTE_PEN_BG_OPACITY</td>
</tr>
<tr>
<td>Indicates a pen back ground opacity attribute of a closed-captioning text.</td>
</tr>
<tr>
<td>static int CC_ATTRIBUTE_PEN_FG_COLOR</td>
</tr>
<tr>
<td>Indicates a pen color attribute to draw closed-captioning text.</td>
</tr>
<tr>
<td>static int CC_ATTRIBUTE_PEN_FG_OPACITY</td>
</tr>
<tr>
<td>Indicates a pen opacity attribute of a closed-captioning text.</td>
</tr>
<tr>
<td>static int CC_ATTRIBUTE_PEN_SIZE</td>
</tr>
<tr>
<td>Indicates a font size attribute of a closed-captioning text.</td>
</tr>
<tr>
<td>static int CC_ATTRIBUTE_WINDOW_BORDER_COLOR</td>
</tr>
<tr>
<td>Indicates a border color attribute of a closed-captioning window.</td>
</tr>
<tr>
<td>static int CC_ATTRIBUTE_WINDOW_BORDER_TYPE</td>
</tr>
<tr>
<td>Indicates a border type attribute of a closed-captioning window.</td>
</tr>
<tr>
<td>static int CC_ATTRIBUTE_WINDOW_FILL_COLOR</td>
</tr>
<tr>
<td>Indicates a window fill color attribute of a closed-captioning window.</td>
</tr>
<tr>
<td>static int CC_ATTRIBUTE_WINDOW_FILL_OPACITY</td>
</tr>
<tr>
<td>Indicates a border type attribute of a closed-captioning window.</td>
</tr>
<tr>
<td>static int CC_BORDER_DEPRESSED</td>
</tr>
<tr>
<td>Indicates a border type of DEPRESSED.</td>
</tr>
<tr>
<td>static int CC_BORDER_NONE</td>
</tr>
<tr>
<td>Indicates a border type of NONE.</td>
</tr>
</tbody>
</table>
### Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int</td>
<td>CC BORDER RAISED</td>
<td>Indicates a border type of RAISED.</td>
</tr>
<tr>
<td>static int</td>
<td>CC BORDER SHADOW LEFT</td>
<td>Indicates a border type of SHADOW_LEFT.</td>
</tr>
<tr>
<td>static int</td>
<td>CC BORDER SHADOW RIGHT</td>
<td>Indicates a border type of SHADOW_RIGHT.</td>
</tr>
<tr>
<td>static int</td>
<td>CC BORDER UNIFORM</td>
<td>Indicates a border type of UNIFORM.</td>
</tr>
<tr>
<td>static int</td>
<td>CC_OPACITY_FLASH</td>
<td>Indicates an opacity value for a flash.</td>
</tr>
<tr>
<td>static int</td>
<td>CC_OPACITY SOLID</td>
<td>Indicates an opacity value for a solid.</td>
</tr>
<tr>
<td>static int</td>
<td>CC_OPACITY TRANSLUCENT</td>
<td>Indicates an opacity value for a translucent.</td>
</tr>
<tr>
<td>static int</td>
<td>CC_OPACITY TRANSPARENT</td>
<td>Indicates an opacity value for a transparent.</td>
</tr>
<tr>
<td>static int</td>
<td>CC PEN SIZE LARGE</td>
<td>Indicates a large pen size.</td>
</tr>
<tr>
<td>static int</td>
<td>CC PEN SIZE SMALL</td>
<td>Indicates a small pen size.</td>
</tr>
<tr>
<td>static int</td>
<td>CC PEN SIZE STANDARD</td>
<td>Indicates a standard pen size.</td>
</tr>
<tr>
<td>static int</td>
<td>CC_TYPE ANALOG</td>
<td>Indicates an analog type closed-captioning.</td>
</tr>
<tr>
<td>static int</td>
<td>CC_TYPE DIGITAL</td>
<td>Indicates a digital type closed-captioning.</td>
</tr>
</tbody>
</table>

### Constructor Summary

<table>
<thead>
<tr>
<th>Access Modifier</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected</td>
<td>ClosedCaptioningAttribute()</td>
<td>A constructor of this class.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Return Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.lang.Object</td>
<td>getCCAttribute(int attribute, int ccType)</td>
<td>This method returns a current attribute values applied to a closed-captioning text on a screen.</td>
</tr>
<tr>
<td>java.lang.Object[]</td>
<td>getCCCapability(int attribute, int ccType)</td>
<td>This method returns a possible attribute values applied to an closed-captioning text on a screen.</td>
</tr>
<tr>
<td>static ClosedCaptioningAttribute</td>
<td>getInstance()</td>
<td>This method returns an instance of this class.</td>
</tr>
</tbody>
</table>
Method Summary

```java
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.
```

Methods inherited from class java.lang.Object

```java
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.

See Also:
Constant Field Values

**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.

See Also:
Constant Field Values

**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.

See Also:
Constant Field Values

**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.

See Also:
Constant Field Values

**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.

See Also:
Constant Field Values

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.
  See Also:
  Constant Field Values

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.
  See Also:
  Constant Field Values

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.
  See Also:
  Constant Field Values

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.
  See Also:
  Constant Field Values

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.
  See Also:
  Constant Field Values

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.
  See Also:
  Constant Field Values
**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.

See Also:
Constant Field Values

**CC_PEN_SIZE_SMALL**

public static final int CC_PEN_SIZE_SMALL

Indicates a small pen size.

See Also:
Constant Field Values

**CC_PEN_SIZE_STANDARD**

public static final int CC_PEN_SIZE_STANDARD

Indicates a standard pen size.

See Also:
Constant Field Values

**CC_PEN_SIZE_LARGE**

public static final int CC_PEN_SIZE_LARGE

Indicates a large pen size.

See Also:
Constant Field Values

**CC_OPACITY_SOLID**

public static final int CC_OPACITY_SOLID

Indicates a opacity value for a solid.

See Also:
Constant Field Values

**CC_OPACITY_FLASH**

public static final int CC_OPACITY_FLASH

Indicates a opacity value for a flash.

See Also:
Constant Field Values

**CC_OPACITY_TRANSLUCENT**

public static final int CC_OPACITY_TRANSLUCENT

Indicates a opacity value for a translucent.

See Also:
Constant Field Values

**CC_OPACITY_TRANSPARENT**

public static final int CC_OPACITY_TRANSPARENT

Indicates a opacity value for a transparent.
See Also:
Constant Field Values

**CC_BORDER_NONE**

public static final int **CC_BORDER_NONE**
Indicates a border type of NONE.
See Also:
Constant Field Values

**CC_BORDER_RAISED**

public static final int **CC_BORDER_RAISED**
Indicates a border type of RAISED.
See Also:
Constant Field Values

**CC_BORDER_DEPRESSED**

public static final int **CC_BORDER_DEPRESSED**
Indicates a border type of DEPRESSED.
See Also:
Constant Field Values

**CC_BORDER_UNIFORM**

public static final int **CC_BORDER_UNIFORM**
Indicates a border type of UNIFORM.
See Also:
Constant Field Values

**CC_BORDER_SHADOW_LEFT**

public static final int **CC_BORDER_SHADOW_LEFT**
Indicates a border type of SHADOW_LEFT.
See Also:
Constant Field Values

**CC_BORDER_SHADOW_RIGHT**

public static final int **CC_BORDER_SHADOW_RIGHT**
Indicates a border type of SHADOW_RIGHT.
See Also:
Constant Field Values

**CC_TYPE_ANALOG**

public static final int **CC_TYPE_ANALOG**
Indicates an analog type closed-captioning.
See Also:
Constant Field Values

**CC_TYPE_DIGITAL**

public static final int **CC_TYPE_DIGITAL**
Indicates an digital type closed-captioning.

See Also:
Constant Field Values

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.
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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CC_ANALOG_SERVICE_CC1</strong></td>
<td>Indicates an analog closed-captioning service CC1.</td>
</tr>
<tr>
<td><strong>CC_ANALOG_SERVICE_CC2</strong></td>
<td>Indicates an analog closed-captioning service CC2.</td>
</tr>
<tr>
<td><strong>CC_ANALOG_SERVICE_CC3</strong></td>
<td>Indicates an analog closed-captioning service CC3.</td>
</tr>
<tr>
<td><strong>CC_ANALOG_SERVICE_CC4</strong></td>
<td>Indicates an analog closed-captioning service CC4.</td>
</tr>
<tr>
<td><strong>CC_ANALOG_SERVICE_T1</strong></td>
<td>Indicates an analog closed-captioning service T1.</td>
</tr>
<tr>
<td><strong>CC_ANALOG_SERVICE_T2</strong></td>
<td>Indicates an analog closed-captioning service T2.</td>
</tr>
<tr>
<td><strong>CC_ANALOG_SERVICE_T3</strong></td>
<td>Indicates an analog closed-captioning service T3.</td>
</tr>
<tr>
<td><strong>CC_ANALOG_SERVICE_T4</strong></td>
<td>Indicates an analog closed-captioning service T4.</td>
</tr>
<tr>
<td><strong>CC_NO_SERVICE</strong></td>
<td>Indicates no closed-captioning service.</td>
</tr>
<tr>
<td><strong>CC_TURN_OFF</strong></td>
<td>Indicates turn digital/analog closed-captioning on.</td>
</tr>
<tr>
<td><strong>CC_TURN_ON</strong></td>
<td>Indicates turn digital/analog closed-captioning off.</td>
</tr>
<tr>
<td><strong>CC_TURN_ON_MUTE</strong></td>
<td>Indicates turn digital/analog closed-captioning on only when muting an audio.</td>
</tr>
</tbody>
</table>
## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void addClosedCaptioningListener(ClosedCaptioningListener ccListener)</code></td>
<td>Add a listener to notify a closed-captioning state change.</td>
</tr>
<tr>
<td><code>int[] getClosedCaptioningServiceNumber()</code></td>
<td>This method returns a current closed-captioning service for a JMF Player that is controlled by a ClosedCaptioningControl instance.</td>
</tr>
<tr>
<td><code>int getClosedCaptioningState()</code></td>
<td>Get the current state of closed-captioning of a JMF Player that is controlled by a ClosedCaptioningControl instance.</td>
</tr>
<tr>
<td><code>int[] getSupportedClosedCaptioningServiceNumber()</code></td>
<td>This method returns closed-captioning service numbers that are supported by a JMF Player that is controlled by a ClosedCaptioningControl instance.</td>
</tr>
<tr>
<td><code>void removeClosedCaptioningListener(ClosedCaptioningListener ccListener)</code></td>
<td>Remove the given ClosedCaptioningListener.</td>
</tr>
<tr>
<td><code>void setClosedCaptioningServiceNumber(int analogServiceNumber, int digitalServiceNumber)</code></td>
<td>This method sets a new closed-captioning service number to be represented by a JMF Player that is controlled by a ClosedCaptioningControl instance.</td>
</tr>
<tr>
<td><code>void setClosedCaptioningState(int turnOn)</code></td>
<td>Turn closed-captioning of a JMF Player that is controlled by a ClosedCaptioningControl instance on or off.</td>
</tr>
</tbody>
</table>

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.  
See Also:  
Constant Field Values

### CC_ANALOG_SERVICE_CC2

`static final int CC_ANALOG_SERVICE_CC2`  
Indicates an analog closed-captioning service CC2.  
See Also:  
Constant Field Values

### CC_ANALOG_SERVICE_CC3

`static final int CC_ANALOG_SERVICE_CC3`  
Indicates an analog closed-captioning service CC3.  
See Also:  
Constant Field Values
CC_ANALOG_SERVICE_CC4
static final int CC_ANALOG_SERVICE_CC4
    Indicates an analog closed-captioning service CC4.
    See Also:
    Constant Field Values

CC_ANALOG_SERVICE_T1
static final int CC_ANALOG_SERVICE_T1
    Indicates an analog closed-captioning service T1.
    See Also:
    Constant Field Values

CC_ANALOG_SERVICE_T2
static final int CC_ANALOG_SERVICE_T2
    Indicates an analog closed-captioning service T2.
    See Also:
    Constant Field Values

CC_ANALOG_SERVICE_T3
static final int CC_ANALOG_SERVICE_T3
    Indicates an analog closed-captioning service T3.
    See Also:
    Constant Field Values

CC_ANALOG_SERVICE_T4
static final int CC_ANALOG_SERVICE_T4
    Indicates an analog closed-captioning service T4.
    See Also:
    Constant Field Values

CC_NO_SERVICE
static final int CC_NO_SERVICE
    Indicates no closed-captioning service.
    See Also:
    Constant Field Values

CC_TURN_OFF
static final int CC_TURN_OFF
    Indicates turn digital/analog closed-captioning on.
    See Also:
    Constant Field Values

CC_TURN_ON
static final int CC_TURN_ON
    Indicates turn digital/analog closed-captioning off.
    See Also:
    Constant Field Values
CC_TURN_ON_MUTE
static final int CC_TURN_ON_MUTE
    Indicates turn digital/analog closed-captioning on only when muting an audio.
    See Also:
    Constant Field Values

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 an 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 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. A value of CC_NO_SERVICE 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. A value of CC_NO_SERVICE 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

```java
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 service. The first item shall be an analog captioning
service number and the second item shall be a digital captioning service number. The array length shall be
2. A value of CC_NO_SERVICE 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

```java
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 numbers 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 numbers 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

```java
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:**
cClickListener - 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").

```java
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:**
cClickListener - a ClosedCaptioningListener instance to be removed.

**Throws:**
java.lang.SecurityException - if the caller doesn't have MonitorAppPermission("handler.closedCaptioning").
org.ocap.media

Class ClosedCaptioningEvent

class ClosedCaptioningEvent
extends java.util.EventObject

This class is an event to notify a change of a closed-captioning state.

See Also:
        Serialized Form

Field Summary

<table>
<thead>
<tr>
<th>static int</th>
<th>EVENTID_CLOSED_CAPTIONING_OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This event indicates a current closed-captioning state is turning off.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>EVENTID_CLOSED_CAPTIONING_ON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This event indicates a current closed-captioning state is turning on.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>EVENTID_CLOSED_CAPTIONING_ON_MUTE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This event indicates a current closed-captioning state is &quot;on mute&quot;.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>EVENTID_CLOSED_CAPTIONING_SELECT_NEW_SERVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This event indicates a new closed-captioning service (C1 to C4, T1 to T4) is selected.</td>
</tr>
</tbody>
</table>

Fields inherited from class java.util.EventObject

source

Constructor Summary

ClosedCaptioningEvent(java.lang.Object source, int id)
Construct a new ClosedCaptioningEvent with the specified event ID.

Method Summary

int getEventID()
Get the event ID associated with this ClosedCaptioningEvent.

Methods inherited from class java.util.EventObject

ggetSource, 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.

See Also:
Constant Field Values

EVENTID_CLOSED_CAPTIONING_OFF
public static final int EVENTID_CLOSED_CAPTIONING_OFF
This event indicates a current closed-captioning state is turning off.

See Also:
Constant Field Values

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".

See Also:
Constant Field Values

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.

See Also:
Constant Field Values

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.

Method Summary

<table>
<thead>
<tr>
<th>Method Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void ccStatusChanged(ClosedCaptioningEvent event)</td>
<td>This method shall be called when a closed-captioning state or a captioning service of the JMF Player that is controlled by the ClosedCaptioningControl instance has changed.</td>
</tr>
</tbody>
</table>

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 controlled by the ClosedCaptioningControl instance has changed. Note that this method is not called when an existence 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.

See Also:
- ClosedCaptioningEvent
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 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.

See Also:
  Serialized Form

### Field Summary

<table>
<thead>
<tr>
<th>Fields inherited from class java.util.EventObject</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
</tr>
</tbody>
</table>

### Constructor Summary

FilterResourceAvailableEvent(VBIFilterGroup f)
A constructor of this class.

### Method Summary

<table>
<thead>
<tr>
<th>java.lang.Object</th>
<th>getSource()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This method returns an instance of a class implementing VBIFilterGroup that is the source of the event.</td>
</tr>
</tbody>
</table>

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:

ggetSource in class 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 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.

See Also:
    Serialized Form

Field Summary

Fields inherited from class java.util.EventObject
source

Constructor Summary

ForcedDisconnectedEvent(VBIFilterGroup f)
A constructor of this class.

Method Summary

java.lang.Object getSource()
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:

gSource in class 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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>getDenialReasons</strong>&lt;br&gt;int</td>
<td>Return a bit mask of denial reasons for the given service component.</td>
</tr>
<tr>
<td><strong>getDeniedElementaryStreams</strong>&lt;br&gt;java.util.Enumeration</td>
<td>Returns the list of service components whose presentation has not been authorized by the MediaAccessHandler.</td>
</tr>
<tr>
<td><strong>isFullAuthorization</strong>&lt;br&gt;boolean</td>
<td>Returns true if the presentation of all service components is authorized.</td>
</tr>
</tbody>
</table>

### Method Detail

**isFullAuthorization**

boolean isFullAuthorization()

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.

**getDenialReasons**

int getDenialReasons(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.
org.ocap.media

**Interface MediaAccessConditionControl**

**All Superinterfaces:**
- javax.media.Control

```java
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 `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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void conditionHasChanged(MediaPresentationEvaluationTrigger trigger)</td>
<td>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.</td>
</tr>
</tbody>
</table>

### Method Detail

**conditionHasChanged**

```java
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 launching or not applications.

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>checkMediaAccessAuthorization</td>
</tr>
<tr>
<td>MediaAccessAuthorization</td>
</tr>
<tr>
<td>checkMediaAccessAuthorization</td>
</tr>
<tr>
<td>(javax.media.Player p, OcapLocator sourceURL, boolean isSourceDigital, ElementaryStream[] esList, MediaPresentationEvaluationTrigger evaluationTrigger)</td>
</tr>
<tr>
<td>The checkMediaAccessAuthorization() method is invoked each time a MediaPresentationEvaluationTrigger is generated either by the OCAP implementation, or, by a monitor application that has MonitorAppPermission(&quot;mediaAccess&quot;) through the MediaAccessConditionControl JMF control.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>checkMediaAccessAuthorization</td>
</tr>
<tr>
<td>MediaAccessAuthorization</td>
</tr>
<tr>
<td>checkMediaAccessAuthorization</td>
</tr>
<tr>
<td>(javax.media.Player p, OcapLocator sourceURL, boolean isSourceDigital, ElementaryStream[] esList, MediaPresentationEvaluationTrigger evaluationTrigger)</td>
</tr>
<tr>
<td>The checkMediaAccessAuthorization() method is invoked each time a MediaPresentationEvaluationTrigger is generated either by the OCAP implementation, or, by a monitor application that has MonitorAppPermission(&quot;mediaAccess&quot;) 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.</td>
</tr>
</tbody>
</table>
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() |
| Constructor of MediaAccessHandlerRegistrar. |

Method Summary

| static MediaAccessHandlerRegistrar getInstance() |
| This method returns the sole instance of the MediaAccessHandlerRegistrar class. |

| abstract void registerMediaAccessHandler(MediaAccessHandler mah) |
| Registers the handler that can prevent the presentation of A/V service components. |

| abstract void setExternalTriggers(MediaPresentationEvaluationTrigger[] triggers) |
| Defines the list of triggers that the Monitor Application wants to generate. |

Methods inherited from class java.lang.Object

close, 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

<table>
<thead>
<tr>
<th>Static</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MediaPresentationEvaluationTrigger</td>
<td>CURRENT_PROGRAM_EVENT_CHANGED MediaPresentationEvaluationTrigger indicating that current program event has changed.</td>
</tr>
<tr>
<td>MediaPresentationEvaluationTrigger</td>
<td>NEW_SELECTED_SERVICE MediaPresentationEvaluationTrigger indicating that a new service has been selected.</td>
</tr>
<tr>
<td>MediaPresentationEvaluationTrigger</td>
<td>NEW_SELECTED_SERVICE_COMPONENTS MediaPresentationEvaluationTrigger indicating that new service components have been selected via JMF control or via ServiceContext.</td>
</tr>
<tr>
<td>MediaPresentationEvaluationTrigger</td>
<td>PMT_CHANGED MediaPresentationEvaluationTrigger indicating that the broadcast PMT has changed.</td>
</tr>
<tr>
<td>MediaPresentationEvaluationTrigger</td>
<td>POWER_STATE_CHANGED MediaPresentationEvaluationTrigger indicating that the power state has changed, e.g., switch to Software Standby.</td>
</tr>
<tr>
<td>MediaPresentationEvaluationTrigger</td>
<td>PROGRAM_EVENT_RATING_CHANGED MediaPresentationEvaluationTrigger indicating that program event rating has changed.</td>
</tr>
</tbody>
</table>
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.

See Also:
   Serialized Form

Field Summary

Fields inherited from class java.util.EventObject
source

Constructor Summary

protected MediaPresentationEvent(javax.media.Controller from, int previous, int current, int target)
   Constructor of MediaPresentationEvent

Method Summary

ElementaryStream[] getPresentedStreams()

OcapLocator getSourceURL()

MediaPresentationEvaluationTrigger getTrigger()

boolean isSourceDigital()
Methods inherited from class javax.media.TransitionEvent
getCurrentState, getPreviousState, getTargetState

Methods inherited from class javax.media.ControllerEvent
ggetSource, getSourceController

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

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 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.

gTrigger

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)
  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()
  Get a first time that was set to this MediaTimer object.

javax.media.Time
  getLastTime()
  Get a last time that was set to this MediaTimer object.

javax.media.Player
  getPlayer()
  Get a Player that was tied to this MediaTimer object by a constructor.

void
  setFirstTime(javax.media.Time time)
  Set a first time of a time range.

void
  setLastTime(javax.media.Time time)
  Set a last time of a time range.

void
  start()
  Start this MediaTimer object.

void
  stop()
  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**

```java
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

All Superinterfaces:
java.util.EventListener

public interface MediaTimerListener
extends java.util.EventListener

This is a listener to inform events on a MediaTimer object.

Field Summary

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMER_START</td>
<td>static int</td>
<td>Indicates a MediaTimer starts.</td>
</tr>
<tr>
<td>TIMER_STOP</td>
<td>static int</td>
<td>Indicates a MediaTimer stops.</td>
</tr>
<tr>
<td>TIMER_WENTOFF_FIRST</td>
<td>static int</td>
<td>Indicates a MediaTimer went off since a current media time passes the specified first time in a reverse playback or a skip playback.</td>
</tr>
<tr>
<td>TIMER_WENTOFF_LAST</td>
<td>static int</td>
<td>Indicates a MediaTimer went off since a current media time passes the specified last time in a normal playback or a skip playback.</td>
</tr>
</tbody>
</table>

Method Summary

void notify(int event, javax.media.Player p)

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.

See Also:
Constant Field Values

TIMER_STOP

static final int TIMER_STOP
Indicates a MediaTimer stops.

See Also:
Constant Field Values

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.

See Also:
Constant Field Values

**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.

See Also:
Constant Field Values

### Method Detail

#### notify

void notify(int event,
  javax.media.Player p)

This is a callback 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.

See Also:
   Serialized Form

Field Summary

Fields inherited from class java.util.EventObject
source

Constructor Summary

protected NormalMediaPresentationEvent (javax.media.Controller from, int previous, int current, int target)
   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

Methods inherited from class javax.media.ControllerEvent
getSource, getSourceController

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

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ElementaryStream[]</td>
<td>getNotPresentedStreams()</td>
</tr>
<tr>
<td>int</td>
<td>getReason(ElementaryStream es)</td>
</tr>
</tbody>
</table>

Method Detail

getNotPresentedStreams

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.

gReason

int getReason(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 S3DConfiguration

public interface S3DConfiguration

This interface represents a 3D frame packing payload as defined in [OCCEP].

Field Summary

<table>
<thead>
<tr>
<th>static int</th>
<th>S3D_AVC_SEI_PAYLOAD_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AVC SEI payload type for 3D frame packing arrangement defined in [OCCEP].</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>S3D_MPEG2_USER_DATA_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MPEG-2 S3D signaling data defined in [OCCEP].</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>int</th>
<th>getDataType()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the data type of the 3D content.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getFormatType()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the 3D content format type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>byte[]</th>
<th>getPayload()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the payload of the 3DTV information description message.</td>
</tr>
</tbody>
</table>

Field Detail

S3D_MPEG2_USER_DATA_TYPE

static final int S3D_MPEG2_USER_DATA_TYPE
MPEG-2 S3D signaling data defined in [OCCEP].
See Also:
Constant Field Values

S3D_AVC_SEI_PAYLOAD_TYPE

static final int S3D_AVC_SEI_PAYLOAD_TYPE
AVC SEI payload type for 3D frame packing arrangement defined in [OCCEP].
See Also:
Constant Field Values

Method Detail

dataType

int getDataType()
Gets the data type of the 3D content. Returns S3D_MPEG2_USER_DATA_TYPE when the stream type is MPEG2_VIDEO, or S3D_AVC_SEI_PAYLOAD_TYPE when the stream type is AVC_VIDEO. Note: other data types may be added in the future.
Returns:
The data type of the 3D content.

getFormatType
int getFormatType()
   Gets the 3D content format type. See S3DFormatTypes for possible return values.
   Returns:
The 3D content format type.

getPayload
byte[] getPayload()
   Gets the payload of the 3DTV information description message. The byte array format will match the
definition for the data type returned by the getDataType method.
org.ocap.media

Interface S3DFormatTypes

public interface S3DFormatTypes

The interface contains constants representing video formats for stereoscopic 3D streams.

Note: Constants can be added from multiple standards for format types supported by tru2way.

Field Summary

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int FORMAT_2D</td>
<td>Video format is 2D rather than 3D.</td>
<td></td>
</tr>
<tr>
<td>static int SIDE_BY_SIDE</td>
<td>S3D frame packing arrangement is Side by Side; see [OCCEP].</td>
<td></td>
</tr>
<tr>
<td>static int TOP_AND_BOTTOM</td>
<td>S3D frame packing arrangement is Top and Bottom; see [OCCEP].</td>
<td></td>
</tr>
<tr>
<td>static int UNKNOWN</td>
<td>Video format is unknown.</td>
<td></td>
</tr>
</tbody>
</table>

Field Detail

UNKNOWN

static final int UNKNOWN
    Video format is unknown.

    See Also:
    Constant Field Values

FORMAT_2D

static final int FORMAT_2D
    Video format is 2D rather than 3D.

    See Also:
    Constant Field Values

SIDE_BY_SIDE

static final int SIDE_BY_SIDE
    S3D frame packing arrangement is Side by Side; see [OCCEP].

    See Also:
    Constant Field Values

TOP_AND_BOTTOM

static final int TOP_AND_BOTTOM
    S3D frame packing arrangement is Top and Bottom; see [OCCEP].

    See Also:
    Constant Field Values
org.ocap.media

Class S3DSignalingChangedEvent

java.lang.Object
  java.util.EventObject
    org.dvb.media.VideoFormatEvent
      org.ocap.media.S3DSignalingChangedEvent

All Implemented Interfaces:
  java.io.Serializable

public class S3DSignalingChangedEvent
  extends VideoFormatEvent

This class represents an event that will be reported to an application with an
org.ocap.media.VideoFormatControl. For a presenting Player the implementation SHALL monitor
the signaling of 3D formatting data, as defined by [OCCEP], and generate this event when:

- 3D formatting data is signaled after presentation starts,
- 3D formatting data changes,
- 3D formatting data is no longer signaled.

See Also:
  VideoFormatControl, Serialized Form

Field Summary

<table>
<thead>
<tr>
<th>static int</th>
<th>TRANSITION_FROM_2D_TO_3D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3D formatting data in content transitioned from no 3D formatting data present (i.e., 2D content), to 3D formatting data present in content.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>TRANSITION_FROM_3D_TO_2D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3D formatting data in content transitioned from 3D formatting data present in content to no 3D formatting data present in content (i.e., 2D content).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>TRANSITION_OF_3D_FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3D formatting data in content transitioned from one format to another; e.g., Side by Side to Top and Bottom, Top and Bottom to Side by Side.</td>
</tr>
</tbody>
</table>

Fields inherited from class java.util.EventObject

source

Constructor Summary

S3DSignalingChangedEvent(javax.media.Player source, int transitionType, S3DConfiguration config)
  Constructs an event.
### Method Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3DConfiguration</td>
<td><code>getConfig()</code></td>
<td>Gets the <code>config</code> value passed to the constructor.</td>
</tr>
<tr>
<td></td>
<td><code>getTransitionType()</code></td>
<td>Gets the <code>transitionType</code> value passed to the constructor.</td>
</tr>
</tbody>
</table>

### Field Detail

#### TRANSITION_FROM_2D_TO_3D

```java
public static final int TRANSITION_FROM_2D_TO_3D
```

3D formatting data in content transitioned from no 3D formatting data present (i.e., 2D content), to 3D formatting data present in content.

See Also:
Constant Field Values

#### TRANSITION_FROM_3D_TO_2D

```java
public static final int TRANSITION_FROM_3D_TO_2D
```

3D formatting data in content transitioned from 3D formatting data present in content to no 3D formatting data present in content (i.e., 2D content).

See Also:
Constant Field Values

#### TRANSITION_OF_3D_FORMAT

```java
public static final int TRANSITION_OF_3D_FORMAT
```

3D formatting data in content transitioned from one format to another; e.g., Side by Side to Top and Bottom, Top and Bottom to Side by Side.

See Also:
Constant Field Values

### Constructor Detail

```java
public S3DSignalingChangedEvent(javax.media.Player source,
int transitionType,
S3DConfiguration config)
```

Constructs an event.

Parameters:
- `source` - The source of the event.
transitionType - Indicates the type of content format change. When the content type transitions from 2D to 3D this parameter is set to TRANSITION_FROM_2D_TO_3D. When the content type transitions from 3D to 2D this parameter is set to TRANSITION_FROM_3D_TO_2D. When the content type transitions between 3D formats this parameter is set to TRANSITION_OF_3D_FORMAT.

cfg - The 3D configuration that was signaled. The value SHALL be null when 2D content is currently signaled.

Method Detail

getTransitionType

public int getTransitionType()

Gets the transitionType value passed to the constructor.

Returns:
The 3D signaling transition type.

getConfig

public S3DConfiguration getConfig()

Gets the config value passed to the constructor.

Returns:
The signaled 3D configuration, or null if 2D content is currently signaled. Note: Rapid changes in 3D signaling may cause the returned S3DConfiguration object to be stale as soon as this method completes.
**org.ocap.media**

**Interface VBIFilter**

```java
public interface VBIFilter
```

This class represents a VBI filter. 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.

See also the VBIFilterGroup.

### Method Summary

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>addVBIFilterListener</code></td>
<td>Add a new VBIFilterListener instance to this VBI filter.</td>
</tr>
<tr>
<td><code>clearBuffer</code></td>
<td>Clear an internal buffer to store retrieved VBI data.</td>
</tr>
<tr>
<td><code>getVBIData</code></td>
<td>This method returns multiple VBI data unit bytes.</td>
</tr>
<tr>
<td><code>removeVBIFilterListener</code></td>
<td>Remove an existing VBIFilterListener instance from this VBI filter.</td>
</tr>
<tr>
<td><code>setNotificationByDataUnits</code></td>
<td>Set the number of data units to receive a cyclic notification.</td>
</tr>
<tr>
<td><code>setNotificationByTime</code></td>
<td>Set a notification time.</td>
</tr>
<tr>
<td><code>setTimeOut</code></td>
<td>Set a timeout value.</td>
</tr>
</tbody>
</table>
### Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th><code>startFiltering(java.lang.Object appData)</code></th>
<th>Initiate filtering of VBI data for the specified line and the specified data format by a VBIFilterGroup.</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>startFiltering(java.lang.Object appData, int offset, byte[] posFilterDef, byte[] posFilterMask, byte[] negFilterDef, byte[] negFilterMask)</code></td>
<td>Initiate filtering of VBI data for the specified line and the specified data format by a VBIFilterGroup.</td>
</tr>
<tr>
<td>void</td>
<td><code>stopFiltering()</code></td>
<td>Stop current filtering of this VBI filter.</td>
</tr>
</tbody>
</table>

### Method Detail

**startFiltering**

```java
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**

```java
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 occurred.

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 receive a cyclic notification. By setting the number of data units, the OCAP implementation notifies a VBIFilterEvent with EVENT_CODE_UNITS_NOTIFICATION cyclically every time 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 affect 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**

```java
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., an incomplete data unit may return. When this method is called, the internal buffer is cleared once, i.e., the next call returns the next byte of retrieved VBI data.

**Returns:**
a concatenated VBI data of the form specified by a `VBIFilterGroup.newVBIFilter(int[], int, int, int, int)` method.

**clearBuffer**

```java
void clearBuffer()
```

Clear an internal buffer to store retrieved VBI data. An application shall call this method before the data buffer is 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 |
|           | Indicates an internal buffer is full.
| static int | EVENT_CODE_FAILED_TO_DESCRAMBLE |
|           | Indicates descrambling is unavailable for current video.
| static int | EVENT_CODE_FIRST_VBI_DATA_AVAILABLE |
|           | Indicates that the first VBI data unit is available.
| static int | EVENT_CODE_FORCIBLE_TERMINATED |
|           | Indicates current filtering is terminated forcibly for any reason except other EVENT_CODE_ constants.
| static int | EVENT_CODE_TIME_NOTIFICATION |
|           | Indicates that a specified time-period elapsed after receiving the first byte of a data unit.
| static int | EVENT_CODE_TIMEOUT |
|           | Indicates a timeout (specified by VBIFilter.setTimeOut(long)) occurred, i.e., this event code indicates no data unit is available.
| static int | EVENT_CODE_UNITS_NOTIFICATION |
|           | Indicates that the specified number of new data units are filtered and stored in a buffer cyclically.
| static int | EVENT_CODE_VIDEO_SOURCE_CHANGED |
|           | Indicates the current video for VBI data unit filtering has changed.

Constructor Summary

VBIFilterEvent()
   Constructor of this class.

Method Summary

java.lang.Object | getAppData()
| This method returns application specific data that was specified by VBIFilter.startFiltering() methods.
**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int getEventCode()</code></td>
<td>This method returns the specific event code that caused this event.</td>
</tr>
<tr>
<td><code>java.lang.Object getSource()</code></td>
<td>This method returns an instance of a class implementing VBIFilter that is the source of the event.</td>
</tr>
</tbody>
</table>

**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**

```java
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.

See Also:
- Constant Field Values

**EVENT_CODE_FORCIBLE_TERMINATED**

```java
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.

See Also:
- Constant Field Values

**EVENT_CODE_VIDEO_SOURCE_CHANGED**

```java
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.

See Also:
- Constant Field Values

**EVENT_CODE_FAILED_TO_DESCRAMBLE**

```java
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.

See Also:
- Constant Field Values

**EVENT_CODE_TIMEOUT**

```java
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.
See Also:
Constant Field Values

EVENT_CODE_BUFFER_FULL
public static final int EVENT_CODE_BUFFER_FULL
Indicates an internal buffer is full. Filtering stops automatically.
See Also:
Constant Field Values

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.
See Also:
Constant Field Values

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.
See Also:
Constant Field Values

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:
an 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:
   ResourceProxy, ResourceServer

public class VBIFilterGroup
extends java.lang.Object
implements ResourceProxy, 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 by using separated fields (specify either field 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 code (EOC, RCL, RDC, RU2, RU3 or RU4). Note that Text service data unit returned from the VBIFilter doesn't contain these text mode in/out codes, but contains 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 make a separate 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. Filtering success is not guaranteed even if the OCAP implementation supports the UNKNOWN format, since an analog waveform is not standardized. The bits in one VBI line and field is a data unit for UNKNOWN format.

Support of the other VBI lines is 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

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int</td>
<td>FIELD_1</td>
</tr>
<tr>
<td>static int</td>
<td>FIELD_2</td>
</tr>
<tr>
<td>static int</td>
<td>FIELD_MIXED</td>
</tr>
<tr>
<td>static int</td>
<td>VBI_DATA_FORMAT_AMOL_I</td>
</tr>
<tr>
<td>static int</td>
<td>VBI_DATA_FORMAT_AMOL_II</td>
</tr>
<tr>
<td>static int</td>
<td>VBI_DATA_FORMAT_EIA608B</td>
</tr>
<tr>
<td>static int</td>
<td>VBI_DATA_FORMAT_NABTS</td>
</tr>
<tr>
<td>static int</td>
<td>VBI_DATA_FORMAT_T1</td>
</tr>
<tr>
<td>static int</td>
<td>VBI_DATA_FORMAT_T2</td>
</tr>
<tr>
<td>static int</td>
<td>VBI_DATA_FORMAT_T3</td>
</tr>
<tr>
<td>static int</td>
<td>VBI_DATA_FORMAT_T4</td>
</tr>
<tr>
<td>static int</td>
<td>VBI_DATA_FORMAT_UNKNOWN</td>
</tr>
<tr>
<td>static int</td>
<td>VBI_DATA_FORMAT_XDS</td>
</tr>
</tbody>
</table>
### Constructor Summary

**VBIFilterGroup**(int numberOfFilters)

Creates a VBIFilterGroup instance that includes the specified number of VBI filters.

### Method Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>addResourceStatusEventListener</code> (ResourceStatusListener listener)</td>
<td>Add a listener object to be notified of changes in the status of resources related to VBI filters.</td>
</tr>
<tr>
<td>void</td>
<td><code>attatch</code> (javax.tv.service.selection.ServiceContext serviceContext, ResourceClient client, java.lang.Object requestData)</td>
<td>This method attempts to reserve all VBI filters held by this VBIFilterGroup, and attaches to a specified ServiceContext.</td>
</tr>
<tr>
<td>void</td>
<td><code>detach</code>()</td>
<td>Releases all filters in this VBIFilterGroup and terminate the connection to the ServiceContext.</td>
</tr>
<tr>
<td>ResourceClient</td>
<td><code>getClient</code>()</td>
<td>Returns the ResourceClient object specified in the attach() method.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>getMixedFilteringCapability</code> (int[] lineNumber, int dataFormat)</td>
<td>Returns true if field mixed filtering of the specified VBI line numbers with the specified data format is supported by this VBIFilterGroup.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>getSCTE20Capability</code>()</td>
<td>Indicates if the host supports line 21 VBI data retrieval from user_data in MPEG picture headers as defined in ANSI/SCTE 20.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>getSCTE21Capability</code>()</td>
<td>Indicates if the host supports line 21 VBI data retrieval from user_data in MPEG picture headers as defined in ANSI/SCTE 21.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>getSeparatedFilteringCapability</code> (int[] lineNumber, int dataFormat)</td>
<td>Returns true if field separated filtering of the specified VBI line numbers with the specified data format is supported by this VBIFilterGroup.</td>
</tr>
<tr>
<td>VBIFilter</td>
<td><code>newVBIFilter</code> (int[] lineNumber, int field, int dataFormat, int unitLength, int bufferSize)</td>
<td>Create a new VBIFilter instance within a VBIFilterGroup instance.</td>
</tr>
<tr>
<td>void</td>
<td><code>removeResourceStatusEventListener</code> (ResourceStatusListener listener)</td>
<td>Remove a specified listener object from this VBIFilterGroup.</td>
</tr>
</tbody>
</table>
Field Detail

VBI_DATA_FORMAT_XDS

public static final int VBI_DATA_FORMAT_XDS
    Represents a XDS data format of line 21 field 2.
    See Also:
    Constant Field Values

VBI_DATA_FORMAT_T1

public static final int VBI_DATA_FORMAT_T1
    Represents a Text service T1 format of line 21 field 1.
    See Also:
    Constant Field Values

VBI_DATA_FORMAT_T2

public static final int VBI_DATA_FORMAT_T2
    Represents a Text service T2 format of line 21 field 1.
    See Also:
    Constant Field Values

VBI_DATA_FORMAT_T3

public static final int VBI_DATA_FORMAT_T3
    Represents a Text service T3 format of line 21 field 2.
    See Also:
    Constant Field Values

VBI_DATA_FORMAT_T4

public static final int VBI_DATA_FORMAT_T4
    Represents a Text service T4 format of line 21 field 2.
    See Also:
    Constant Field Values

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.
    See Also:
    Constant Field Values

VBI_DATA_FORMAT_NABTS

public static final int VBI_DATA_FORMAT_NABTS
    Represents an NABTS data format defined by EIA-516.
    See Also:
Constant Field Values

**VBI_DATA_FORMAT_AMOL_I**

```java
public static final int VBI_DATA_FORMAT_AMOL_I
    Represents an AMOL I data format defined by ACN 403-1218-024.
    See Also:
    Constant Field Values
```

**VBI_DATA_FORMAT_AMOL_II**

```java
public static final int VBI_DATA_FORMAT_AMOL_II
    Represents an AMOL II data format defined by ACN 403-1218-024.
    See Also:
    Constant Field Values
```

**VBI_DATA_FORMAT_UNKNOWN**

```java
public static final int VBI_DATA_FORMAT_UNKNOWN
    Represents an unknown. I.e., all data bits in the VBI line are retrieved.
    See Also:
    Constant Field Values
```

**FIELD_1**

```java
public static final int FIELD_1
    Represents filtering of VBI data in field 1 only.
    See Also:
    Constant Field Values
```

**FIELD_2**

```java
public static final int FIELD_2
    Represents filtering of VBI data in field 2 only.
    See Also:
    Constant Field Values
```

**FIELD_MIXED**

```java
public static final int FIELD_MIXED
    Represents filtering of VBI data in both field 1 and field 2 in arrival order.
    See Also:
    Constant Field Values
```

---

**Constructor Detail**

**VBIFilterGroup**

```java
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 formats.
bufferSize - number of bytes to specify size of an internal buffer used to hold a VBI data unit.

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 number specified by the 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,
    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.

detach

public void detach()
Releases 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.

ggetServiceContext

public javax.tv.service.selection.ServiceContext getServiceContext()

Returns a ServiceContext instance specified by a

Returns:
a ServiceContext instance specified by a

ggetSeparatedFilteringCapability

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.

ggetMixedFilteringCapability

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.

ggetClient
public ResourceClient getClient()
Returns the ResourceClient object specified in the attach() method.

Specified by:
ggetClient in interface 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(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 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(ResourceStatusListener listener)
Remove a specified listener object from this VBIFilterGroup.

**Specified by:**
removeResourceStatusEventListener in interface 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

All Superinterfaces:
    java.util.EventListener

public interface VBIFilterListener
extends java.util.EventListener

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void filterUpdate</td>
<td>This method is called by the OCAP implementation to notify an application that a VBI event has occurred.</td>
</tr>
</tbody>
</table>

### 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 an event code.
org.oicap.media

Interface VideoFormatControl

All Superinterfaces:
javax.media.Control, VideoFormatControl

public interface VideoFormatControl
extends VideoFormatControl

This interface extends org.dvb.media.VideoFormatControl to provide access to OCAP-specific info signaled in presented video, such as 3D formatting data.

All instances of VideoFormatControl returned from calls to Controller.getControl(java.lang.String) or Controller.getControls() within an OCAP implementation SHALL also be instances of org.oicap.media.VideoFormatControl.

Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCANMODE_INTERLACED</td>
<td>Constant</td>
</tr>
<tr>
<td>SCANMODE.Progressive</td>
<td>Constant</td>
</tr>
<tr>
<td>SCANMODE_UNKNOWN</td>
<td>Constant</td>
</tr>
</tbody>
</table>

Fields inherited from interface org.dvb.media.VideoFormatControl

AFD_14_9, AFD_14_9.TOP, AFD_16_9, AFD_16_9.SP_14_9, AFD_16_9.SP_4_3,
AFD_16_9.TOP, AFD_4_3, AFD_4_3.SP_14_9, AFD_GT_16_9, AFD_NOT_PRESENT,
AFD_SAME, ASPECT_RATIO_16_9, ASPECT_RATIO_2_21_1, ASPECT_RATIO_4_3,
ASPECT_RATIO_UNKNOWN, DAR_16_9, DAR_4_3, DFCPLATFORM,
DFC_PROCESSING_16_9.ZOOM, DFC_PROCESSING_CCO, DFC_PROCESSING_FULL,
DFC_PROCESSING_LB_14_9, DFC_PROCESSING_LB_16_9,
DFC_PROCESSING_LB_2_21_1.ON_16_9, DFC_PROCESSING_LB_2_21_1.ON_4_3,
DFC_PROCESSING_NONE, DFC_PROCESSING_PAN_SCAN, DFC_PROCESSING_UNKNOWN

Method Summary

S3DConfiguration()  
getS3DConfiguration()  
Returns the 3D configuration info of the video.

Methods inherited from interface org.dvb.media.VideoFormatControl

addVideoFormatListener, getActiveFormatDefinition, getAspectRatio,
getDecoderFormatConversion, getDisplayAspectRatio, getVideoTransformation,
isPlatform, removeVideoFormatListener

Methods inherited from interface javax.media.Control

getcControlComponent
Field Detail

SCANMODE_UNKNOWN
static final int SCANMODE_UNKNOWN
    Constant indicating an unknown or unspecified line scan mode.
    See Also:
    Constant Field Values

SCANMODE_INTERLACED
static final int SCANMODE_INTERLACED
    Constant indicating interlaced line scan mode.
    See Also:
    Constant Field Values

SCANMODE_PROGRESSIVE
static final int SCANMODE_PROGRESSIVE
    Constant indicating progressive line scan mode.
    See Also:
    Constant Field Values

Method Detail

getS3DConfiguration
S3DConfiguration getS3DConfiguration()
    Returns the 3D configuration info of the video. See [OCCEP] for the 3D formatting data definition. Returns null if no 3D formatting data is present (e.g., in the case of 2D video). Note: Rapid changes in 3D signaling may cause the returned S3DConfiguration object to be stale as soon as this method completes.
    Returns:
    The signaled 3D formatting data, or null if no 3D formatting data is present.

getScanMode
int getScanMode()
    Reports the scan mode of the input video. A value of SCANMODE_UNKNOWN MAY be returned, indicating that the scan mode is unknown or unspecified.
    Returns:
    one of SCANMODE_UNKNOWN, SCANMODE_INTERLACED, or SCANMODE_PROGRESSIVE.
Annex T  OCAP 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 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-MHP 1.0.3], which is not included in [DVB-GEM 1.0.2]. OCAP does not use any of the APIs specified in this section of the [DVB-MHP 1.0.3].

Table T–1 - Correlation between OCAP and [DVB-GEM 1.0.2] and [DVB-MHP 1.0.3]

<table>
<thead>
<tr>
<th>OCAP</th>
<th>[DVB-GEM 1.0.2] Section</th>
<th>GEM Compliance</th>
<th>[DVB-MHP 1.0.3] Section</th>
<th>MHP Compliance</th>
</tr>
</thead>
</table>

T.2  OCAP Specific Requirements

This information extends the specification requirements made to the [DVB-MHP 1.0.3].

T.2.1  Extensions to DVB-MHP (Normative)

The following information is normative for OCAP. It is specific to OCAP and is not contained within the [DVB-MHP 1.0.3].

The Service Information used in systems supporting OCAP is more closely aligned with that used in SCTE than in DVB. Consequently, OCAP will follow this SI model. 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. 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 SI Packages

The following packages are contained in multiple specifications including the [DVB-MHP 1.0.3], and OCAP:

- 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:

- 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.

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 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 the CableCARD device present and describes variations associated with the different [SCTE 65] profiles.

**T.2.1.2.1  JavaTV Abstract SI API Considerations Specific to OCAP**

[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 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

<table>
<thead>
<tr>
<th>JavaTV SI Method</th>
<th>Suggested Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>javax.tv.media.MediaSelectControl</td>
<td>The implementation SHOULD throw InvalidLocatorException if any of: select(Locator), select(Locator []), or replace(Locator, Locator) methods is called with an OOB OcapLocator. Analog and digital services can be selected as described in the select() and replace() methods for MediaSelectControl.</td>
</tr>
<tr>
<td>javax.tv.service.selection.ServiceContext</td>
<td>The implementation SHOULD throw InvalidLocatorException if the select(Locator []) method is called with an OOB OcapLocator argument.</td>
</tr>
<tr>
<td>javax.tv.service.SIManager</td>
<td>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.</td>
</tr>
</tbody>
</table>

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

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 Network Text Tables

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.2.2.1.3 Short-form Virtual Channel Table

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.2.2.1.4 Long-form Virtual Channel Table

For Profile 6, 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.2.2.1.5 Event Information Tables

For Profiles 4 through 6, 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 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, an empty String object with 0 length is returned. The javadoc for this method in [Java TV] specifies the return value is the string representation of the service number if the service name is not available. This section asserts the service number is the empty string for purposes of the getName method.

**T.2.2.3.2 getServiceType**

When the LVCT 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

<table>
<thead>
<tr>
<th>Service Type Code</th>
<th><code>javax.tv.Service.ServiceType</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>ANALOG_TV</td>
</tr>
<tr>
<td>0x02</td>
<td>DIGITAL_TV</td>
</tr>
<tr>
<td>0x03</td>
<td>DIGITAL_RADIO</td>
</tr>
<tr>
<td>0x04</td>
<td>DATA_BROADCAST</td>
</tr>
</tbody>
</table>

### 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 is provided, and specifies a two-part channel number, the `minor_channel_number` is returned. When the LVCT is provided, and specifies a one-part channel number, -1 is returned. When the LVCT is not provided, but the optional Two Part Channel Number descriptor is provided in the SVCT, the `minor_channel_number` is returned. When the LVCT is 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 is provided, and specifies a two-part channel number, the `major_channel_number` is returned. When the LVCT is provided, and specifies a one-part channel number, the `one_part_number` is returned. When the LVCT is not provided, but the optional Two Part Channel Number descriptor is provided in the SVCT, the `major_channel_number` is returned. When the LVCT is 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, -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. 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 General Preference 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-C] 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 the V-chip is always working independently of the OCAP implementation, so that the service will be stopped by the V-chip module. However, if the application wants to block the service based on proprietary rating 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-C] for US rating dimension or Symbol described in Section 6.3 of [CEA-766-C]. 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 ProgramScheduleListener 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>
<thead>
<tr>
<th>Stream type value</th>
<th>Java TV Stream Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>UNKNOWN</td>
</tr>
<tr>
<td>0x01</td>
<td>VIDEO</td>
</tr>
<tr>
<td>0x02</td>
<td>VIDEO</td>
</tr>
<tr>
<td>Stream type value</td>
<td>Java TV Stream Type</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>0x03</td>
<td>AUDIO</td>
</tr>
<tr>
<td>0x04</td>
<td>AUDIO</td>
</tr>
<tr>
<td>0x05</td>
<td>SECTIONS</td>
</tr>
<tr>
<td>0x06</td>
<td>DATA</td>
</tr>
<tr>
<td>0x07</td>
<td>UNKNOWN</td>
</tr>
<tr>
<td>0x08</td>
<td>DATA</td>
</tr>
<tr>
<td>0x09</td>
<td>UNKNOWN</td>
</tr>
<tr>
<td>0x0A-0x0D</td>
<td>DATA</td>
</tr>
<tr>
<td>0x0E</td>
<td>UNKNOWN</td>
</tr>
<tr>
<td>0x0F</td>
<td>AUDIO</td>
</tr>
<tr>
<td>0x10</td>
<td>UNKNOWN</td>
</tr>
<tr>
<td>0x11</td>
<td>AUDIO</td>
</tr>
<tr>
<td>0x12-0x13</td>
<td>UNKNOWN</td>
</tr>
<tr>
<td>0x14</td>
<td>SECTIONS</td>
</tr>
<tr>
<td>0x15</td>
<td>DATA</td>
</tr>
<tr>
<td>0x16-0x19</td>
<td>SECTIONS</td>
</tr>
<tr>
<td>0x1A</td>
<td>UNKNOWN</td>
</tr>
<tr>
<td>0x1B</td>
<td>VIDEO</td>
</tr>
<tr>
<td>0x1C-0x7F</td>
<td>UNKNOWN</td>
</tr>
<tr>
<td>0x80</td>
<td>VIDEO</td>
</tr>
<tr>
<td>0x81</td>
<td>AUDIO</td>
</tr>
<tr>
<td>0x82-0x86</td>
<td>UNKNOWN</td>
</tr>
<tr>
<td>0x87</td>
<td>AUDIO</td>
</tr>
<tr>
<td>0x88-0xFF</td>
<td>UNKNOWN</td>
</tr>
</tbody>
</table>

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 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, 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 retrieveComponents

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  \textit{javax.tv.service.navigation.ServiceProviderInformation}

This interface is not implemented in OCAP.

T.2.2.18  \textit{javax.tv.service.transport.Bouquet}

This interface is not implemented in OCAP.

T.2.2.19  \textit{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  \textit{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.

\textit{T.2.2.20.1 getDeliverySystemType}

This method always returns a DeliverySystemType of CABLE.

\textit{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  \textit{javax.tv.service.transport.TransportStream}

This interface provides basic information about the transport stream.

\textit{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 the TSID is not available from either the LVCT or SVCT, then this method SHALL use the PAT to obtain the TSID. In this scenario, if the TSID is not available when this method is called, then this method SHALL block for 500 milliseconds or until the PAT is available, whichever comes first. If the TSID is not available from any source, then this method SHALL return 0xFFFF. This method SHALL return 0xFFFF for an analog source.

\textit{T.2.2.21.2 getDescription}

When available, this method returns the short_name from the LVCT. If not available, returns an empty string.

T.2.2.22  \textit{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.2.3  Mapping of the DAVIC SI API to OpenCable SI**

T.2.3.1  *org.davic.mpeg.TransportStream*

T. 2. 3. 1. 1  *getTransportStreamID*

This method SHALL behave as defined by Annex T.2.2.21.1
Package org.ocap.si

The org.ocap.si package represents MPEG-2 PSI data.

See:
- Description

### Interface Summary

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DescriptorTag</strong></td>
<td>The DescriptorTag contains constant values to be read from the descriptor_tag field in a Descriptor.</td>
</tr>
<tr>
<td><strong>PATProgram</strong></td>
<td>This interface represents an program block in the information loop of the MPEG-2 PAT. Each PAT will contain a loop of these blocks.</td>
</tr>
<tr>
<td><strong>PMTElementaryStreamInfo</strong></td>
<td>This interface represents an MPEG-2 PMT Elementary Stream Info loop.</td>
</tr>
<tr>
<td><strong>ProgramAssociationTable</strong></td>
<td>This interface represents an MPEG-2 Program Association Table (PAT).</td>
</tr>
<tr>
<td><strong>ProgramMapTable</strong></td>
<td>This interface represents an MPEG-2 Program Map Table (PMT).</td>
</tr>
<tr>
<td><strong>StreamType</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>This interface represents an MPEG-2 Program Specific Information (PSI) table structure.</td>
</tr>
<tr>
<td><strong>TableChangeListener</strong></td>
<td>This interface is implemented by an application for notification of change to a table.</td>
</tr>
</tbody>
</table>

### Class Summary

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Descriptor</strong></td>
<td>This class represents an MPEG-2 descriptor.</td>
</tr>
<tr>
<td><strong>ProgramAssociationTableManager</strong></td>
<td>The Program Association Table (PAT) manager is used to discover and listen for PATs.</td>
</tr>
<tr>
<td><strong>ProgramMapTableManager</strong></td>
<td>The Program Map Table (PMT) manager is used to discover and listen for MPEG-2 PMTs.</td>
</tr>
</tbody>
</table>

### Package org.ocap.si Description

The org.ocap.si package represents MPEG-2 PSI data.
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

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descriptor()</td>
<td>public</td>
</tr>
</tbody>
</table>

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract byte getByteAt(int index)</td>
<td>public abstract</td>
<td>Get a particular byte within the descriptor content.</td>
</tr>
<tr>
<td>abstract byte[] getContent()</td>
<td>public abstract</td>
<td>Get the data contained within this descriptor.</td>
</tr>
<tr>
<td>abstract short getContentLength()</td>
<td>public abstract</td>
<td>Get the descriptor_length field.</td>
</tr>
<tr>
<td>abstract short getTag()</td>
<td>public abstract</td>
<td>Get the descriptor_tag field.</td>
</tr>
</tbody>
</table>

## Methods inherited from class java.lang.Object

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait</td>
<td>public</td>
<td></td>
</tr>
</tbody>
</table>

## 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

```java
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

```java
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

```java
public abstract byte getByteAt(int index)
```

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**

```java
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**

<table>
<thead>
<tr>
<th>static short</th>
<th>value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC3_AUDIO</td>
<td>static short 10</td>
<td>SCTE AC-3_audio_descriptor.</td>
</tr>
<tr>
<td>APPLICATION</td>
<td>static short 11</td>
<td>MHP application_descriptor.</td>
</tr>
<tr>
<td>APPLICATION_ICONS</td>
<td>static short 12</td>
<td>MHP application_icons_descriptor.</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>static short 13</td>
<td>MHP application_name_descriptor.</td>
</tr>
<tr>
<td>APPLICATION_SIGNALING</td>
<td>static short 14</td>
<td>MHP application_signaling_descriptor.</td>
</tr>
<tr>
<td>ASSOCIATION_TAG</td>
<td>static short 19</td>
<td>DSMCC association_tag_descriptor</td>
</tr>
<tr>
<td>ATSC_PRIVATE_INFORMATION</td>
<td>static short 20</td>
<td>ATSC private_information_descriptor.</td>
</tr>
<tr>
<td>AUDIO_STREAM</td>
<td>static short 21</td>
<td>MPEG-2 audio_stream_descriptor.</td>
</tr>
<tr>
<td>CA</td>
<td>static short 22</td>
<td>MPEG-2 CA_descriptor.</td>
</tr>
<tr>
<td>CACHING_PRIORITY</td>
<td>static short 23</td>
<td>MHP caching_priority_descriptor.</td>
</tr>
<tr>
<td>CAPTION_SERVICE</td>
<td>static short 24</td>
<td>SCTE caption_service_descriptor.</td>
</tr>
<tr>
<td>CAROUSEL_IDENTIFIER</td>
<td>static short 25</td>
<td>DSMCC carousel_identifier_descriptor.</td>
</tr>
<tr>
<td>CHANNEL_PROPERTIES</td>
<td>static short 26</td>
<td>SCTE channel_properties_descriptor.</td>
</tr>
<tr>
<td>COMPONENT_NAME</td>
<td>static short 27</td>
<td>SCTE component_name_descriptor.</td>
</tr>
<tr>
<td>CONTENT_ADVISORY</td>
<td>static short 28</td>
<td>SCTE content_advisory_descriptor.</td>
</tr>
<tr>
<td>CONTENT_TYPE</td>
<td>static short 29</td>
<td>MHP content_type_descriptor.</td>
</tr>
</tbody>
</table>
### Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static short</td>
<td><strong>COPYRIGHT</strong> &lt;br&gt;MPEG-2 copyright_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>DATA_broadcast_ID</strong> &lt;br&gt;MHP data_broadcast_id_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>DATA_stream_ALIGNMENT</strong> &lt;br&gt;MPEG-2 data_stream_alignment_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>DAYLIGHT_SAVINGS_TIME</strong> &lt;br&gt;SCTE daylight_savings_time_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>DII_LOCATION</strong> &lt;br&gt;MHP DII_location_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>DVB_J_APPLICATION</strong> &lt;br&gt;MHP DVB-J_application_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>DVB_J_APPLICATION_LOCATION</strong> &lt;br&gt;MHP DVB-J_application_location_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>EXTENDED_CHANNEL_NAME_DESCRIPTION</strong> &lt;br&gt;SCTE extended_channel_name_description_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>EXTERNAL_APPLICATION_AUTHORISATION</strong> &lt;br&gt;MHP external_application_authorisation_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>HIERARCHY</strong> &lt;br&gt;MPEG-2 hierarchy_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>IBP</strong> &lt;br&gt;MPEG-2 IBP_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>IPV4_ROUTING</strong> &lt;br&gt;MHP IPV4_routing_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>IPV6_ROUTING</strong> &lt;br&gt;MHP IPV6_routing_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>ISO_639_LANGUAGE</strong> &lt;br&gt;MPEG-2 ISO_639_language_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>LABEL</strong> &lt;br&gt;MHP label_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>MAC_ADDRESS_LIST</strong> &lt;br&gt;MAC address list</td>
</tr>
<tr>
<td>static short</td>
<td><strong>MAXIMUM_BITRATE</strong> &lt;br&gt;MPEG-2 maximum_bitrate_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>MULTIPLEX_UTILIZATION_BUFFER</strong> &lt;br&gt;MPEG-2 multiplex_utilization_buffer_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>PRE_FETCH</strong> &lt;br&gt;MHP pre-fetch_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>PRIVATE_DATA_INDICATOR</strong> &lt;br&gt;MPEG-2 private_data_indicator_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>PRIVATE_DATASpecifier</strong> &lt;br&gt;MHP private_data_specifier_descriptor.</td>
</tr>
</tbody>
</table>
Field Summary

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static short</td>
<td><strong>REGISTRATION</strong> MPEG-2 registration_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>REVISION_DETECTION</strong> SCTE revision_detection_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>SERVICE_IDENTIFIER</strong> MHP service_identifier_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>SERVICE_LOCATION</strong> SCTE service_location_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>SMOOTHING_BUFFER</strong> MPEG-2 smoothing_buffer_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>STD</strong> MPEG-2 STD_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>STREAM_IDENTIFIER</strong> DVB stream_identifier_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>STUFFING</strong> SCTE stuffing_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>SYSTEM_CLOCK</strong> MPEG-2 system_clock_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>TARGET_BACKGROUND_GRID</strong> MPEG-2 target_background_grid Descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>TIME_SHIFTED_SERVICE</strong> SCTE time_shifted_service_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>TRANSPORT_PROTOCOL</strong> MHP transport_protocol_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>TWO_PART_CHANNEL_NUMBER</strong> SCTE two_part_channel_number_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>VIDEO_STREAM</strong> MPEG-2 video_stream_descriptor.</td>
</tr>
<tr>
<td>static short</td>
<td><strong>VIDEO_WINDOW</strong> MPEG-2 video_window_descriptor.</td>
</tr>
</tbody>
</table>

Field Detail

**VIDEO_STREAM**

static final short **VIDEO_STREAM**

MPEG-2 video_stream_descriptor.

See Also:

Constant Field Values

**AUDIO_STREAM**

static final short **AUDIO_STREAM**

MPEG-2 audio_stream_descriptor.

See Also:
Constant Field Values

HIERARCHY
static final short HIERARCHY
  MPEG-2 hierarchy_descriptor.
  See Also:
  Constant Field Values

REGISTRATION
static final short REGISTRATION
  MPEG-2 registration_descriptor.
  See Also:
  Constant Field Values

DATA_STREAM_ALIGNMENT
static final short DATA_STREAM_ALIGNMENT
  MPEG-2 data_stream_alignment_descriptor.
  See Also:
  Constant Field Values

TARGET_BACKGROUND_GRID
static final short TARGET_BACKGROUND_GRID
  MPEG-2 target_background_grid_descriptor.
  See Also:
  Constant Field Values

VIDEO_WINDOW
static final short VIDEO_WINDOW
  MPEG-2 video_window_descriptor.
  See Also:
  Constant Field Values

CA
static final short CA
  MPEG-2 CA_descriptor.
  See Also:
  Constant Field Values

ISO_639_LANGUAGE
static final short ISO_639_LANGUAGE
  MPEG-2 ISO_639_language_descriptor.
  See Also:
  Constant Field Values

SYSTEM_CLOCK
static final short SYSTEM_CLOCK
  MPEG-2 system_clock_descriptor.
See Also:
Constant Field Values

MULTIPLEX_UTILIZATION_BUFFER
static final short MULTIPLEX_UTILIZATION_BUFFER
   MPEG-2 multiplex_utilization_buffer_descriptor.
   See Also:
       Constant Field Values

COPYRIGHT
static final short COPYRIGHT
   MPEG-2 copyright_descriptor.
   See Also:
       Constant Field Values

MAXIMUM_BITRATE
static final short MAXIMUM_BITRATE
   MPEG-2 maximum_bitrate_descriptor.
   See Also:
       Constant Field Values

PRIVATE_DATA_INDICATOR
static final short PRIVATE_DATA_INDICATOR
   MPEG-2 private_data_indicator_descriptor.
   See Also:
       Constant Field Values

SMOOTHING_BUFFER
static final short SMOOTHING_BUFFER
   MPEG-2 smoothing_buffer_descriptor.
   See Also:
       Constant Field Values

STD
static final short STD
   MPEG-2 STD_descriptor.
   See Also:
       Constant Field Values

IBP
static final short IBP
   MPEG-2 IBP_descriptor.
   See Also:
       Constant Field Values

CAROUSEL_IDENTIFIER
static final short CAROUSEL_IDENTIFIER
DSMCC carousel_identifier_descriptor

See Also:
Constant Field Values

ASSOCIATION_TAG

static final short ASSOCIATION_TAG
   DSMCC association_tag_descriptor

See Also:
Constant Field Values

APPLICATION

static final short APPLICATION
   MHP application_descriptor.

See Also:
Constant Field Values

APPLICATION_NAME

static final short APPLICATION_NAME
   MHP application_name_descriptor.

See Also:
Constant Field Values

TRANSPORT_PROTOCOL

static final short TRANSPORT_PROTOCOL
   MHP transport_protocol_descriptor.

See Also:
Constant Field Values

DVB_J_APPLICATION

static final short DVB_J_APPLICATION
   MHP DVB-J_application_descriptor.

See Also:
Constant Field Values

DVB_JAPPLICATION_LOCATION

static final short DVB_JAPPLICATION_LOCATION
   MHP DVB-J_application_location_descriptor.

See Also:
Constant Field Values

EXTERNALAPPLICATION_AUTHORISATION

static final short EXTERNALAPPLICATION_AUTHORISATION
   MHP external_application_authorisation_descriptor.

See Also:
Constant Field Values
IPV4_ROUTING
static final short IPV4_ROUTING
    MHP IPV4_routing_descriptor.
    See Also:
    Constant Field Values

IPV6_ROUTING
static final short IPV6_ROUTING
    MHP IPV6_routing_descriptor.
    See Also:
    Constant Field Values

APPLICATION_ICONS
static final short APPLICATION_ICONS
    MHP application_icons_descriptor.
    See Also:
    Constant Field Values

PRE_FETCH
static final short PRE_FETCH
    MHP pre-fetch_descriptor.
    See Also:
    Constant Field Values

DII_LOCATION
static final short DII_LOCATION
    MHP DII_location_descriptor.
    See Also:
    Constant Field Values

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.
    See Also:
    Constant Field Values

PRIVATE_DATASpecifier
static final short PRIVATE_DATASpecifier
    MHP private_dataSpecifier_descriptor.
    See Also:
    Constant Field Values

DATA.Broadcast_ID
static final short DATA.Broadcast_ID
    MHP data_broadcast_id_descriptor.
    See Also:
Constant Field Values

**APPLICATION_SIGNALING**
static final short APPLICATION_SIGNALING
MHP application_signaling_descriptor.
See Also:
Constant Field Values

**SERVICE_IDENTIFIER**
static final short SERVICE_IDENTIFIER
MHP service_identifier_descriptor.
See Also:
Constant Field Values

**LABEL**
static final short LABEL
MHP label_descriptor.
See Also:
Constant Field Values

**CACHING_PRIORITY**
static final short CACHING_PRIORITY
MHP caching_priority_descriptor.
See Also:
Constant Field Values

**CONTENT_TYPE**
static final short CONTENT_TYPE
MHP content_type_descriptor.
See Also:
Constant Field Values

**STUFFING**
static final short STUFFING
SCTE stuffing_descriptor.
See Also:
Constant Field Values

**AC3_AUDIO**
static final short AC3_AUDIO
SCTE AC-3_audio_descriptor.
See Also:
Constant Field Values

**CAPTION_SERVICE**
static final short CAPTION_SERVICE
SCTE caption_service_descriptor.
See Also:
Constant Field Values

**CONTENT_ADVISORY**

static final short CONTENT_ADVISORY
SCTE content_advisory_descriptor.
See Also:
Constant Field Values

**REVISION_DETECTION**

static final short REVISION_DETECTION
SCTE revision_detection_descriptor.
See Also:
Constant Field Values

**TWO_PART_CHANNEL_NUMBER**

static final short TWO_PART_CHANNEL_NUMBER
SCTE two_part_channel_number_descriptor.
See Also:
Constant Field Values

**CHANNEL_PROPERTIES**

static final short CHANNEL_PROPERTIES
SCTE channel_properties_descriptor
See Also:
Constant Field Values

**DAYLIGHT_SAVINGS_TIME**

static final short DAYLIGHT_SAVINGS_TIME
SCTE daylight_savings_time_descriptor.
See Also:
Constant Field Values

**EXTENDED_CHANNEL_NAME_DESCRIPTION**

static final short EXTENDED_CHANNEL_NAME_DESCRIPTION
SCTE extended_channel_name_description_descriptor.
See Also:
Constant Field Values

**SERVICE_LOCATION**

static final short SERVICE_LOCATION
SCTE service_location_descriptor.
See Also:
Constant Field Values

**TIME_SHIFTED_SERVICE**

static final short TIME_SHIFTED_SERVICE
SCTE time_shifted_service_descriptor.
See Also:
Constant Field Values

COMPONENT_NAME
static final short COMPONENT_NAME
SCTE component_name_descriptor.
See Also:
Constant Field Values

MAC_ADDRESS_LIST
static final short MAC_ADDRESS_LIST
MAC address list
See Also:
Constant Field Values

ATSC_PRIVATE_INFORMATION
static final short ATSC_PRIVATE_INFORMATION
ATSC private_information_descriptor
See Also:
Constant Field Values
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int getPID()</td>
<td>Get the program_map_PID field.</td>
</tr>
<tr>
<td>int getProgramNumber()</td>
<td>Get the program_number field.</td>
</tr>
</tbody>
</table>

### Method Detail

#### getProgramNumber

```java
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 by getPID(), references the network PID.

Returns:
The program number.

#### getPID

```java
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descriptor[] getDescriptorLoop()</td>
<td>Get the descriptors associated with the elementary stream.</td>
</tr>
<tr>
<td>short getElementaryPID()</td>
<td>Get the elementary_PID field.</td>
</tr>
<tr>
<td>java.lang.String getLocatorString()</td>
<td>Get the locator for the elementary stream.</td>
</tr>
<tr>
<td>short getStreamType()</td>
<td>Get the stream_type field.</td>
</tr>
</tbody>
</table>

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.

get Elementary PID

short getElementaryPID()

Get the elementary_PID field. Thirteen bit field specifying the PID of the associated elementary stream.

Returns:
The elementary PID.

descriptor Loop

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

```java
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PATProgram[] getPrograms()</code></td>
<td>Get the program loop in Program Association Table.</td>
</tr>
<tr>
<td><code>short getTableId()</code></td>
<td>Get the identifier for this table.</td>
</tr>
</tbody>
</table>

### Method Detail

**getTableId**

```java
short getTableId()
```

Get the identifier for this table.

**getPrograms**

```java
PATProgram[] getPrograms()
```

Get the program loop in Program Association Table.
Returns:
The list of PATProgram which represents the program loop in Program Association Table.
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**

<table>
<thead>
<tr>
<th>protected</th>
<th>ProgramAssociationTableManager()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For Singleton behavior</td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>abstract void</th>
<th>addInBandChangeListener</th>
</tr>
</thead>
<tbody>
<tr>
<td>(TableChangeListener listener, javax.tv.locator.Locator locator)</td>
<td>Add a TableChangeListener object that will be notified when the inband PAT for the channel (transport stream) identified by the Locator parameter changes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>abstract void</th>
<th>addOutOfBandChangeListener</th>
</tr>
</thead>
<tbody>
<tr>
<td>(TableChangeListener listener)</td>
<td>Add a TableChangeListener object that will be notified when the out-of-band PAT changes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static ProgramAssociationTableManager</th>
<th>getInstance()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Get an instance of the Program Association Table Manager.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>abstract void</th>
<th>removeInBandChangeListener</th>
</tr>
</thead>
<tbody>
<tr>
<td>(TableChangeListener listener)</td>
<td>Remove the TableChangeListener object for the inband PAT.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>abstract void</th>
<th>removeOutOfBandChangeListener</th>
</tr>
</thead>
<tbody>
<tr>
<td>(TableChangeListener listener)</td>
<td>Remove the TableChangeListener object for the OOB PAT.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>abstract javax.tv.service.SIRequest</th>
<th>retrieveInBand</th>
</tr>
</thead>
<tbody>
<tr>
<td>(javax.tv.service.SIRequestor requestor, java.lang.tv.locator.Locator locator)</td>
<td>Retrieve a PAT from the in-band channel (transport stream) identified by the Locator parameter.</td>
</tr>
</tbody>
</table>
### Method Summary

| abstract javax.tv.service.SIRequest retrieveOutOfBand (javax.tv.service.SIRequestor requestor) |
| Retrieve the PAT from the out-of-band channel. |

### Constructor Detail

**ProgramAssociationTableManager**

```java
protected ProgramAssociationTableManager ()
```

For Singleton behavior

### Method Detail

**getInstance**

```java
public static ProgramAssociationTableManager getInstance ()
```

Get an instance of the Program Association Table Manager.

**Returns:**

The ProgramAssociationTableManager instance.

**addInBandChangeListener**

```java
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. `javax.tv.service.SIRequestFailureType.DATA_UNAVAILABLE` is returned via `SIRequestor.notifyFailure()` if the locator refers to an analog service.

**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://n=service_name, ocap://f=frequency.program_number

**Throws:**

- `java.lang.IllegalArgumentException` - This exception is thrown when the `locator` parameter is not in the form of a valid OAP URL as specified by this method.

**addOutOfBandChangeListener**

```java
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 registered SIRequestor.NotifyFailure() method may have expired.
javax.tv.service.SIRequestFailureType.DATA_UNAVAILABLE is returned via
SIRequestor.NotifyFailure() if the locator refers to an analog service.

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://n=service_name,
ocap://f=frequency.program_number

Returns:
The SIRequest object that identifies this asynchronous retrieval request and allows the request to be
cancelled.

Throws:
java.lang.IllegalArgumentException - This exception is thrown when the locator
parameter is not in the form of a valid OCAP URL as specified by this method.

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 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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descriptor[]</td>
<td>getOuterDescriptorLoop() Get the outer descriptor loop.</td>
</tr>
<tr>
<td>int</td>
<td>getPcrPID() Get the PCR_PID field.</td>
</tr>
<tr>
<td>PMTElementaryStreamInfo[]</td>
<td>getPMTElementaryStreamInfoLoop() Get elementary stream information blocks.</td>
</tr>
<tr>
<td>int</td>
<td>getProgramNumber() Get the program_number field, corresponds with the PMT.</td>
</tr>
</tbody>
</table>

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.

```c
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.

```c
Descriptor[] getOuterDescriptorLoop()
```

Get the outer descriptor loop. List of descriptors that pertains to all programs.

**Returns:**
The outer descriptor loop.

```c
PMTElementaryStreamInfo[] getPMTElemantaryStreamInfoLoop()
```

Get elementary stream information blocks. Each block contains elementary stream data for a particular stream type.

**Returns:**
The elementary stream information blocks.
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()  
For Singleton behavior

**Method Summary**

- abstract void **addInBandChangeListener** (TableChangeListener listener, javax.tv.locator.Locator locator)  
  Add a TableChangeListener object that will be notified when the inband PMT changes.

- abstract void **addOutOfBandChangeListener** (TableChangeListener listener, int programNumber)  
  Add a TableChangeListener object that will be notified when the out-of-band PMT changes.

- static ProgramMapTableManager **getInstance**()  
  Get an instance of the Program Map Table Manager.

- abstract void **removeInBandChangeListener** (TableChangeListener listener)  
  Remove the TableChangeListener object for the inband PMT.

- abstract void **removeOutOfBandChangeListener** (TableChangeListener listener)  
  Remove the TableChangeListener object for the OOB PMT.

- 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.
Method Summary

<table>
<thead>
<tr>
<th>Method Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>abstract javax.tv.service.SIRequest retrieveOutOfBand(javax.tv.service.SIRequestor requestor, int programNumber)</code></td>
<td>Retrieve the PMT from the out-of-band channel.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, 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. javax.tv.service.SIRequestFailureType.DATA_UNAVAILABLE is returned via SIRequestor.notifyFailure() if the locator refers to an analog service.

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://n=service_name, ocap://f=frequency.program_number

Throws:
- `java.lang.IllegalArgumentException`: This exception is thrown when the `locator` parameter is not in the form of a valid OCAP URL as specified by this method.

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**

```java
public abstract void removeInBandChangeListener(TableChangeListener listener)
```

Remove the TableChangeLister object for the inband PMT.

**Parameters:**

- **listener** - The TableChangeLister object to be removed.

**removeOutOfBandChangeListener**

```java
public abstract void removeOutOfBandChangeListener(TableChangeListener listener)
```

Remove the TableChangeLister object for the OOB PMT.

**Parameters:**

- **listener** - The TableChangeLister object to be removed.

**retrievelnBand**

```java
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. `javax.tv.service.SIRequestFailureType.DATA_UNAVAILABLE` is returned via `SIRequestor.notifyFailure()` if the locator refers to an analog service.

**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://n=service_name`, `ocap://f=frequency.program_number`

**Returns:**

The SIRequest object that identifies this asynchronous retrieval request and allows the request to be cancelled.

**Throws:**

- `java.lang.IllegalArgumentException` - This exception is thrown when the locator parameter is not in the form of a valid OCAP URL as specified by this method.
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 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.
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.

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static short</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>static short</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>static short</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>static short</td>
</tr>
<tr>
<td>static short</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>static short</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>static short</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>static short</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>static short</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>static short</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>static short</td>
</tr>
<tr>
<td>static short</td>
</tr>
<tr>
<td>static short</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>static short</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>static short</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
## Field Summary

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>static short ISO_14496_VISUAL</strong></td>
<td>ISO/IEC 14496-2 Visual</td>
</tr>
<tr>
<td><strong>static short ISOCHRONOUS_DATA</strong></td>
<td>Isochronous data (Methods for Isochronous Data Services Transport, ANSI/SCTE 19 2006)</td>
</tr>
<tr>
<td><strong>static short METADATA_DATA_CAROUSEL</strong></td>
<td>Metadata carried in ISO/IEC 13818-6 Data Carousel</td>
</tr>
<tr>
<td><strong>static short METADATA_OBJECT_CAROUSEL</strong></td>
<td>Metadata carried in ISO/IEC 13818-6 Object Carousel</td>
</tr>
<tr>
<td><strong>static short METADATA_PES</strong></td>
<td>Metadata carried in PES packets</td>
</tr>
<tr>
<td><strong>static short METADATA_SECTIONS</strong></td>
<td>Metadata carried in metadata_sections</td>
</tr>
<tr>
<td><strong>static short METADATA_SYNCH_DOWNLOAD</strong></td>
<td>Metadata carried in ISO/IEC 13818-6 Synchronized Download Protocol</td>
</tr>
<tr>
<td><strong>static short MHEG</strong></td>
<td>ISO/IEC 13522 MHEG.</td>
</tr>
<tr>
<td><strong>static short MPEG_1_AUDIO</strong></td>
<td>ISO/IEC 11172-3 Audio.</td>
</tr>
<tr>
<td><strong>static short MPEG_1_VIDEO</strong></td>
<td>ISO/IEC 11172-2 Video.</td>
</tr>
<tr>
<td><strong>static short MPEG_2_AUDIO</strong></td>
<td>ISO/IEC 13818-3 Audio.</td>
</tr>
<tr>
<td><strong>static short MPEG_2_VIDEO</strong></td>
<td>ITU-T Rec.</td>
</tr>
<tr>
<td><strong>static short MPEG_PRIVATE_DATA</strong></td>
<td>ITU-T Rec.</td>
</tr>
<tr>
<td><strong>static short MPEG_PRIVATE_SECTION</strong></td>
<td>ITU-T Rec.</td>
</tr>
<tr>
<td><strong>static short STD_SUBTITLE</strong></td>
<td>Standard subtitle.</td>
</tr>
<tr>
<td><strong>static short SYNCHRONIZED_DOWNLOAD</strong></td>
<td>ISO/IEC 13818-6 Synchronized Download Protocol</td>
</tr>
<tr>
<td><strong>static short VIDEO_DCII</strong></td>
<td>DigiCipher II video.</td>
</tr>
</tbody>
</table>

## Field Detail

**MPEG_1_VIDEO**

static final short MPEG_1_VIDEO
ISO/IEC 11172-2 Video.
See Also:
Constant Field Values

MPEG_2_VIDEO
static final short MPEG_2_VIDEO
See Also:
Constant Field Values

MPEG_1_AUDIO
static final short MPEG_1_AUDIO
ISO/IEC 11172-3 Audio.
See Also:
Constant Field Values

MPEG_2_AUDIO
static final short MPEG_2_AUDIO
ISO/IEC 13818-3 Audio.
See Also:
Constant Field Values

MPEG_PRIVATE_SECTION
static final short MPEG_PRIVATE_SECTION
ITU-T Rec. H.222.0 ISO/IEC 13818-1 private-sections.
See Also:
Constant Field Values

MPEG_PRIVATE_DATA
static final short MPEG_PRIVATE_DATA
ITU-T Rec. H.222.0 ISO/IEC 13818-1 PES packets containing private data.
See Also:
Constant Field Values

MHEG
static final short MHEG
ISO/IEC 13522 MHEG.
See Also:
Constant Field Values

DSM_CC
static final short DSM_CC
13818-1 Annex A - DSM CC.
See Also:
Constant Field Values

H_222
static final short H_222
See Also: Constant Field Values

**DSM_CC_MPE**

static final short DSM_CC_MPE
ISO/IEC 13818-6 type A (Multi-protocol Encapsulation).
See Also: Constant Field Values

**DSM_CC_UN**

static final short DSM_CC_UN
See Also: Constant Field Values

**DSM_CC_STREAM_DESCRIPTORS**

static final short DSM_CC_STREAM_DESCRIPTORS
ISO/IEC 13818-6 type C (DSM-CC Stream Descriptors).
See Also: Constant Field Values

**DSM_CC_SECTIONS**

static final short DSM_CC_SECTIONS
ISO/IEC 13818-6 type D (DSM-CC Sections any type, including private data).
See Also: Constant Field Values

**AUXILIARY**

static final short AUXILIARY
ISO/IEC 13818-1 auxiliary.
See Also: Constant Field Values

**AAC_ADTS_AUDIO**

static final short AAC_ADTS_AUDIO
ISO/IEC 13818-7 AAC Audio with ADTS transport syntax
See Also: Constant Field Values

**ISO_14496_VISUAL**

static final short ISO_14496_VISUAL
ISO/IEC 14496-2 Visual
See Also: Constant Field Values
AAC_AUDIO_LATM

static final short AAC_AUDIO_LATM
ISO/IEC 14496-3 and ISO/IEC 13818-7 AAC Audio with the LATM transport syntax as defined in
ISO/IEC 14496-3/AMD-1
See Also:
Constant Field Values

FLEXMUX_PES

static final short FLEXMUX_PES
ISO/IEC 14496-1 SL-packetized stream or FlexMux stream carried in PES packets
See Also:
Constant Field Values

FLEXMUX_SECTIONS

static final short FLEXMUX_SECTIONS
ISO/IEC 14496-1 SL-packetized stream or FlexMux stream carried in ISO/IEC14496_sections
See Also:
Constant Field Values

SYNCHRONIZED_DOWNLOAD

static final short SYNCHRONIZEDDOWNLOAD
ISO/IEC 13818-6 Synchronized Download Protocol
See Also:
Constant Field Values

METADATA_PES

static final short METADATA_PES
Metadata carried in PES packets
See Also:
Constant Field Values

METADATA_SECTIONS

static final short METADATA_SECTIONS
Metadata carried in metadata_sections
See Also:
Constant Field Values

METADATA_DATA_CAROUSEL

static final short METADATA_DATA_CAROUSEL
Metadata carried in ISO/IEC 13818-6 Data Carousel
See Also:
Constant Field Values

METADATA_OBJECT_CAROUSEL

static final short METADATA_OBJECT_CAROUSEL
Metadata carried in ISO/IEC 13818-6 Object Carousel
See Also:
Constant Field Values

**METADATA_SYNCH_DOWNLOAD**

static final short METADATA_SYNCH_DOWNLOAD

Metadata carried in ISO/IEC 13818-6 Synchronized Download Protocol

See Also:

Constant Field Values

**MPEG_2_IPMP**

static final short MPEG_2_IPMP

IPMP stream (defined in ISO/IEC 13818-11, MPEG-2 IPMP)

See Also:

Constant Field Values

**AVC_VIDEO**

static final short AVC_VIDEO

AVC video stream as defined in ITU-T Rec. H.264 | ISO/IEC 14496-10

See Also:

Constant Field Values

**VIDEO_DCII**

static final short VIDEO_DCII

DigiCipher II video.

See Also:

Constant Field Values

**ATSC_AUDIO**

static final short ATSC_AUDIO


See Also:

Constant Field Values

**STD_SUBTITLE**

static final short STD_SUBTITLE

Standard subtitle.

See Also:

Constant Field Values

**ISOCHRONOUS_DATA**

static final short ISOCHRONOUS_DATA

Isochronous data (Methods for Isochronous Data Services Transport, ANSI/SCTE 19 2006)

See Also:

Constant Field Values

**ASYNCHRONOUS_DATA**

static final short ASYNCHRONOUS_DATA

Asynchronous data (Methods for Asynchronous Data Services Transport, ANSI/SCTE 53 2008)
See Also:
Constant Field Values

ENHANCED_ATSC_AUDIO

static final short ENHANCED_ATSC_AUDIO
See Also:
Constant Field Values
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

<table>
<thead>
<tr>
<th>short</th>
<th>getTableId()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the table_id field of the table.</td>
</tr>
</tbody>
</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

```java
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>notifyChange</td>
<td>void notifyChange(javax.tv.service.SIChangeEvent event)</td>
<td>This method notifies that the SI table has changed.</td>
</tr>
</tbody>
</table>

### Method Detail

**notifyChange**

```java
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 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 and [DVB-GEM 1.0.2] |
|-----------------|-----------------|-----------------|
| OCAP            | [DVB-GEM 1.0.2] | GEM Compliance  |
| Annex U OCAP System Event API | No Corresponding Section | OCAP-Specific Extension |

Package org.ocap.system.event

The org.ocap.system.event package defines various system events.

See:

Description

Interface Summary

| SystemEventListener | System event handler implemented by a trusted application and registered with SystemEventManager. |

Class Summary

<table>
<thead>
<tr>
<th>ClassSummary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CableCARDResetEvent</td>
<td>Events that indicate the CableCARD is about to be reset and has successfully completed initialization after a reset.</td>
</tr>
<tr>
<td>DeferredDownloadEvent</td>
<td>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]).</td>
</tr>
<tr>
<td>ErrorEvent</td>
<td>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.</td>
</tr>
<tr>
<td>RebootEvent</td>
<td>This class represents an event returned by the system when a reboot is instigated.</td>
</tr>
<tr>
<td>ResourceDepletionEvent</td>
<td>Event that indicates resources are low, and the system is about to destroy application(s) to attempt to correct this.</td>
</tr>
<tr>
<td>SystemEvent</td>
<td>This class is the basis for system event messages.</td>
</tr>
<tr>
<td>SystemEventManager</td>
<td>Registration mechanism for trusted applications to set the error handler.</td>
</tr>
</tbody>
</table>

Package org.ocap.system.event Description

The org.ocap.system.event package defines various system events.

Following is a Java example demonstrating how an application can register to be the error event handler:

```java
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("EventListenerAppSample.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);
    }
}
org.ocap.system.event
Class CableCARDResetEvent

java.lang.Object
    org.ocap.system.event.SystemEvent
    org.ocap.system.event.CableCARDResetEvent

class CableCARDResetEvent extends SystemEvent

Events that indicate the CableCARD is about to be reset and has successfully completed initialization after a reset.

Field Summary

<table>
<thead>
<tr>
<th>static int</th>
<th>CABLECARD_RESET_BEGIN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CableCARD reset request to Host</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>CABLECARD_RESET_COMPLETE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CableCARD reset and initialization complete</td>
</tr>
</tbody>
</table>

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_CABLECARD_RESET_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_APP_RES_DEP_RESERVED_EVENT_TYPES, END_SYS_CABLECARD_RESET_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

CableCARDResetEvent(int typeCode)
    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

Field Detail

CABLECARD_RESET_BEGIN
public static final int CABLECARD_RESET_BEGIN
   CableCARD reset request to Host
   See Also:
   Constant Field Values

CABLECARD_RESET_COMPLETE
public static final int CABLECARD_RESET_COMPLETE
   CableCARD reset and initialization complete
   See Also:
   Constant Field Values

Constructor Detail

CableCARDResetEvent
public CableCARDResetEvent(int typeCode)
   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. 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.
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_REC_ERROR_RESERVED_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)
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)
    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 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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP_CAT_GENERAL_ERROR</td>
<td>Application catastrophic error that doesn't fit into any other given category.</td>
</tr>
<tr>
<td>APP_INFO_GENERAL_EVENT</td>
<td>Application informational event that doesn't fit into any other given category.</td>
</tr>
<tr>
<td>APP_REC_GENERAL_ERROR</td>
<td>Application recoverable error that doesn't fit into any other given category.</td>
</tr>
<tr>
<td>APP_REC_JAVA_THROWABLE</td>
<td>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.</td>
</tr>
<tr>
<td>SYS_CAT_GENERAL_ERROR</td>
<td>System catastrophic error that doesn't fit into any other given category.</td>
</tr>
<tr>
<td>SYS_CAT_JAVA_THROWABLE</td>
<td>Java Throwable thrown by a call made by an application but not caught by the application.</td>
</tr>
<tr>
<td>SYS_INFO_GENERAL_EVENT</td>
<td>System informational event that doesn't fit into any other given category.</td>
</tr>
<tr>
<td>SYS_REC_GENERAL_ERROR</td>
<td>System error that doesn't fit into any other given category.</td>
</tr>
</tbody>
</table>
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)
  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,
        AppID appId)
  This constructor is provided for internal use by OCAP implementations; applications SHOULD NOT call it.

getErrorEvent(int typeCode, java.lang.Throwable throwable)
  Class constructor specifying the event type code, and throwable condition.

Method Summary
java.lang.String getMessage()
  Gets the readable message String that was passed to a constructor explicitly or within a Throwable object.

java.lang.String getStackTrace()
  Gets the stack trace from the Throwable object if a Throwable object was passed to the appropriate constructor.

java.lang.String[] getThrowableClasses()
  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**

```java
public static final int APP_INFO_GENERAL_EVENT
    Application informational event that doesn't fit into any other given category.
    See Also:
    Constant Field Values
```

**APP_REC_GENERAL_ERROR**

```java
public static final int APP_REC_GENERAL_ERROR
    Application recoverable error that doesn't fit into any other given category.
    See Also:
    Constant Field Values
```

**APP_REC_JAVA_THROWABLE**

```java
public static final int APPREC_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.
    See Also:
    Constant Field Values
```

**APP_CAT_GENERAL_ERROR**

```java
public static final int APP_CAT_GENERAL_ERROR
    Application catastrophic error that doesn't fit into any other given category.
    See Also:
    Constant Field Values
```

**SYS_INFO_GENERAL_EVENT**

```java
public static final int SYS_INFO_GENERAL_EVENT
    System informational event that doesn't fit into any other given category.
    See Also:
    Constant Field Values
```

**SYS_REC_GENERAL_ERROR**

```java
public static final int SYS_REC_GENERAL_ERROR
    System error that doesn't fit into any other given category.
    See Also:
    Constant Field Values
```

**SYS_CAT_GENERAL_ERROR**

```java
public static final int SYS_CAT_GENERAL_ERROR
    System catastrophic error that doesn't fit into any other given category.
    See Also:
    Constant Field Values
```
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.
See Also:
Constant Field Values

Constructor Detail

ErrorEvent

public ErrorEvent(int typeCode,
        java.lang.String message)
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_ERROREVENT_TYPES.

ErrorEvent

public ErrorEvent(int typeCode,
        java.lang.Throwable throwable)
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,
        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 |
| REBOOTedBy implementation; no error encountered. |
| static int | REBOOT_BY_TRUSTED_APP |
| Reboot instigated by trusted application. |
| static int | REBOOT_FOR_UNRECOVERABLE_HY_ERROR |
| Reboot instigated by the implementation; unrecoverable hardware error encountered. |
| static int | REBOOT_FOR_UNRECOVERABLE_SYS_ERROR |
| 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_REC_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_REC_ERROR_EVENT_TYPES, END_SYS_RES_DEP_EVENT_TYPES

Constructor Summary

RebootEvent(int typeCode, java.lang.String message)
System event constructor assigns an eventId, Date, and ApplicationIdentifier.

RebootEvent(int typeCode, java.lang.String message, long date, 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.
See Also: Constant Field Values

REBOOT_FOR_UNRECOVERABLE_SYS_ERROR

public static final int REBOOT_FOR_UNRECOVERABLE_SYS_ERROR
Reboot instigated by implementation; unrecoverable system error encountered.
See Also: Constant Field Values

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.
See Also: Constant Field Values

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)
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,
                    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
| Available CPU cycles is depleted to an implementation specific threshold. |
| static int | RESOURCE_RC_BANDWIDTH_DEPLETED
| Available reverse channel bandwidth is depleted to implementation specific threshold. |
| static int | RESOURCE_SYS_MEM_DEPLETED
| Overall system memory is depleted to an implementation specific threshold. |
| static int | RESOURCE_VM_MEM_DEPLETED
| 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)
System event constructor assigns an eventId, Date, and ApplicationIdentifier.

ResourceDepletionEvent(int typeCode, java.lang.String message, long date, 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

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.
    See Also:
    Constant Field Values

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.
    See Also:
    Constant Field Values

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.
    See Also:
    Constant Field Values

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.
    See Also:
    Constant Field Values
Constructor Detail

ResourceDepletionEvent

public ResourceDepletionEvent(int typeCode,
                               java.lang.String message)

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,
                               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

Direct Known Subclasses:
CableCARDResetEvent, 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.

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int BEGIN_APP_CAT_ERROR_EVENT_TYPES</td>
</tr>
<tr>
<td>static int BEGIN_APP_CAT_ERROR_RESERVED_EVENT_TYPES</td>
</tr>
<tr>
<td>static int BEGIN_APP_INFO_EVENT_TYPES</td>
</tr>
<tr>
<td>static int BEGIN_APP_INFO_RESERVED_EVENT_TYPES</td>
</tr>
<tr>
<td>static int BEGIN_APP_REC_ERROR_EVENT_TYPES</td>
</tr>
<tr>
<td>static int BEGIN_APP_REC_ERROR_RESERVED_EVENT_TYPES</td>
</tr>
<tr>
<td>static int BEGIN_SYS_CABLECARD_RESET_EVENT_TYPES</td>
</tr>
</tbody>
</table>
## Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static int</td>
<td>BEGIN_SYS_CAT_ERROR_EVENT_TYPES</td>
<td>Start of range for system generated catastrophic events.</td>
</tr>
<tr>
<td>Static int</td>
<td>BEGIN_SYS_CAT_ERROR_RESERVED_EVENT_TYPES</td>
<td>Start of range reserved for system generated catastrophic error events defined by OCAP.</td>
</tr>
<tr>
<td>Static int</td>
<td>BEGIN_SYS_DNLD_EVENT_TYPES</td>
<td>Start of range reserved for system generated deferred download events in response to a CVT signaling a deferred download.</td>
</tr>
<tr>
<td>Static int</td>
<td>BEGIN_SYS_INFO_EVENT_TYPES</td>
<td>Start of range for system generated informational events.</td>
</tr>
<tr>
<td>Static int</td>
<td>BEGIN_SYS_INFO_RESERVED_EVENT_TYPES</td>
<td>Start of range reserved for system generated informational events defined by OCAP.</td>
</tr>
<tr>
<td>Static int</td>
<td>BEGIN_SYS_REBOOT_EVENT_TYPES</td>
<td>Start of range for system generated reboot events.</td>
</tr>
<tr>
<td>Static int</td>
<td>BEGIN_SYS_REBOOT_RESERVED_EVENT_TYPES</td>
<td>Start of range reserved for system generated reboot events defined by OCAP.</td>
</tr>
<tr>
<td>Static int</td>
<td>BEGIN_SYS_REC_ERROR_EVENT_TYPES</td>
<td>Start of range for system generated recoverable error events.</td>
</tr>
<tr>
<td>Static int</td>
<td>BEGIN_SYS_REC_ERROR_RESERVED_EVENT_TYPES</td>
<td>Start of range reserved for system generated recoverable error events defined by OCAP.</td>
</tr>
<tr>
<td>Static int</td>
<td>BEGIN_SYS_RES_DEP_EVENT_TYPES</td>
<td>Start of range for system generated resource depletion events.</td>
</tr>
<tr>
<td>Static int</td>
<td>BEGIN_SYS_RES_DEP_RESERVED_EVENT_TYPES</td>
<td>Start of range reserved for system generated resource depletion events defined by OCAP.</td>
</tr>
<tr>
<td>Static int</td>
<td>END_APP_CAT_ERROR_EVENT_TYPES</td>
<td>End of range for application generated catastrophic error events.</td>
</tr>
<tr>
<td>Static int</td>
<td>END_APP_INFO_EVENT_TYPES</td>
<td>End of range for application generated informational events.</td>
</tr>
<tr>
<td>Static int</td>
<td>END_APP_REC_ERROR_EVENT_TYPES</td>
<td>End of range for application generated recoverable error events.</td>
</tr>
<tr>
<td>Static int</td>
<td>END_SYS_CABLECARD_RESET_EVENT_TYPES</td>
<td>End of range for system CableCARD reset events.</td>
</tr>
<tr>
<td>Static int</td>
<td>END_SYS_CAT_ERROR_EVENT_TYPES</td>
<td>End of range for system generated catastrophic error events.</td>
</tr>
<tr>
<td>Static int</td>
<td>END_SYS_DNLD_EVENT_TYPES</td>
<td>End of range for system deferred download events.</td>
</tr>
<tr>
<td>Static int</td>
<td>END_SYS_INFO_EVENT_TYPES</td>
<td>End of range for system generated informational events.</td>
</tr>
<tr>
<td>Static int</td>
<td>END_SYS_REBOOT_EVENT_TYPES</td>
<td>End of range for system generated reboot events.</td>
</tr>
<tr>
<td>Static int</td>
<td>END_SYS_RECV_ERROR_EVENT_TYPES</td>
<td>End of range for system generated recoverable error events.</td>
</tr>
<tr>
<td>Static int</td>
<td>END_SYS_RES_DEP_EVENT_TYPES</td>
<td>End of range for system generated resource depletion events.</td>
</tr>
</tbody>
</table>
### Constructor Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected</td>
<td><code>SystemEvent(int typeCode)</code></td>
<td>System event constructor.</td>
</tr>
<tr>
<td>protected</td>
<td><code>SystemEvent(int typeCode, java.lang.String message)</code></td>
<td>System event constructor with message.</td>
</tr>
<tr>
<td>protected</td>
<td><code>SystemEvent(int typeCode, java.lang.String message, long date, AppID appId)</code></td>
<td>This constructor is provided for internal use by OCAP implementations; applications SHOULD NOT call it.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AppID</code></td>
<td><code>getAppID()</code></td>
<td>Gets the globally unique identifier of the application logging the event.</td>
</tr>
<tr>
<td><code>long</code></td>
<td><code>getDate()</code></td>
<td>Gets the event date in milli-seconds from midnight January 1, 1970, GMT.</td>
</tr>
<tr>
<td><code>java.lang.String</code></td>
<td><code>getMessage()</code></td>
<td>Gets the readable message.</td>
</tr>
<tr>
<td><code>int</code></td>
<td><code>getTypeCode()</code></td>
<td>Gets the event type code.</td>
</tr>
</tbody>
</table>

### Field Detail

**BEGIN_SYS_INFO_EVENT_TYPES**

```java
public static final int BEGIN_SYS_INFO_EVENT_TYPES
```
Start of range for system generated informational events.

**BEGIN_SYS_INFO_RESERVED_EVENT_TYPES**

```java
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**

```java
public static final int END_SYS_INFO_EVENT_TYPES
```
End of range for system generated informational events.

See Also:
Constant Field Values
Constant Field Values

BEGIN_APP_INFO_EVENT_TYPES
public static final int BEGIN_APP_INFO_EVENT_TYPES
Start of range for application generated informational events.
See Also:
Constant Field Values

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.
See Also:
Constant Field Values

END_APP_INFO_EVENT_TYPES
public static final int END_APP_INFO_EVENT_TYPES
End of range for application generated informational events.
See Also:
Constant Field Values

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.
See Also:
Constant Field Values

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.
See Also:
Constant Field Values

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.
See Also:
Constant Field Values

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.

See Also:
Constant Field Values

BEGIN_APP_REC_ERROR_RESERVED_EVENT_TYPES

```java
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.

See Also:
Constant Field Values

END_APP_REC_ERROR_EVENT_TYPES

```java
public static final int END_APP_REC_ERROR_EVENT_TYPES
```
End of range for application generated recoverable error events.

See Also:
Constant Field Values

BEGIN_SYS_CAT_ERROR_EVENT_TYPES

```java
public static final int BEGIN_SYS_CAT_ERROR_EVENT_TYPES
```
Start of range for system generated catastrophic events.

These 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.

See Also:
Constant Field Values

BEGIN_SYS_CAT_ERROR_RESERVED_EVENT_TYPES

```java
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.

See Also:
Constant Field Values

END_SYS_CAT_ERROR_EVENT_TYPES

```java
public static final int END_SYS_CAT_ERROR_EVENT_TYPES
```
End of range for system generated catastrophic error events.

See Also:
Constant Field Values

BEGIN_APP_CAT_ERROR_EVENT_TYPES

```java
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.

See Also:
Constant Field Values
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.
    See Also:
    Constant Field Values

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.
    See Also:
    Constant Field Values

BEGIN_SYS_REBOOT_EVENT_TYPES
public static final int BEGIN_SYS_REBOOT_EVENT_TYPES
    Start of range for system generated reboot events.
    See Also:
    Constant Field Values

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.
    See Also:
    Constant Field Values

END_SYS_REBOOT_EVENT_TYPES
public static final int END_SYS_REBOOT_EVENT_TYPES
    End of range for system generated reboot events.
    See Also:
    Constant Field Values

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.
    See Also:
    Constant Field Values

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.
    See Also:
    Constant Field Values
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.
    See Also:
    Constant Field Values

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.
    See Also:
    Constant Field Values

END_SYS_DNLD_EVENT_TYPES
public static final int END_SYS_DNLD_EVENT_TYPES
    End of range for system deferred download events.
    See Also:
    Constant Field Values

BEGIN_SYS_CABLECARD_RESET_EVENT_TYPES
public static final int BEGIN_SYS_CABLECARD_RESET_EVENT_TYPES
    Start of range reserved for system generated CableCARD reset events. This reserved range ends with END_SYS_CABLECARD_RESET_EVENT_TYPES.
    See Also:
    Constant Field Values

END_SYS_CABLECARD_RESET_EVENT_TYPES
public static final int END_SYS_CABLECARD_RESET_EVENT_TYPES
    End of range for system CableCARD reset events.
    See Also:
    Constant Field Values

Constructor Detail

SystemEvent
protected SystemEvent(int typeCode)
    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)
    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,
    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 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).

g getTypeCode

public int getTypeCode()

Gets the event type code. Identifies a code specific to the event system.

Returns:
type code of the event.

d 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.

g getMessage

public java.lang.String getMessage()

Gets the readable message.

Returns:
message string of the event.
Interface SystemEventListener

All Superinterfaces:
java.util.EventListener

public interface SystemEventListener
extends java.util.EventListener

System event handler implemented by a trusted application and registered with SystemEventManager.

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void notifyEvent(SystemEvent event)</td>
<td>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.</td>
</tr>
</tbody>
</table>

Method Detail

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

class SystemEventManager
extends java.lang.Object

Registration mechanism for trusted applications to set the error handler.

Field Summary

<table>
<thead>
<tr>
<th>static int</th>
<th>CABLE_CARD_EVENT_LISTENER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Identifies the CableCARD reset event listener.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>DEFERRED_DOWNLOAD_EVENT_LISTENER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Identifies the deferred download event listener.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>ERROR_EVENT_LISTENER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Identifies the system error event listener.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>REBOOT_EVENT_LISTENER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Identifies the reboot event listener.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>RESOURCE_DEPLETION_EVENT_LISTENER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Identifies the system resource depletion event listener.</td>
</tr>
</tbody>
</table>

Constructor Summary

protected SystemEventManager()  
This constructor must not be used by OCAP applications.

Method Summary

<table>
<thead>
<tr>
<th>static SystemEventManager</th>
<th>getInstance()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the singleton instance of the system event manager.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>abstract void</th>
<th>log(SystemEvent event)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Logs an event.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>abstract void</th>
<th>setEventListener(int type, SystemEventListener sel)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set the system event listener specified by type and the new handler.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>abstract void</th>
<th>unsetEventListener(int type)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unset the system event handler specified by type.</td>
</tr>
</tbody>
</table>

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), Constant
    Field Values

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), Constant
    Field Values

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), Constant
    Field Values

CABLE_CARD_EVENT_LISTENER

public static final int CABLE_CARD_EVENT_LISTENER
    Identifies the CableCARD reset event listener.
    See Also:
    setEventListener(int, org.ocap.system.event.SystemEventListener), Constant
    Field Values

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)

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: One of ERROR_EVENT_LISTENER, REBOOT_EVENT_LISTENER, RESOURCE_DEPLETION_EVENT_LISTENER, DEFERRED_UPLOAD_EVENT_LISTENER, or CABLE_CARD_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, DEFERRED_UPLOAD_EVENT_LISTENER, or CABLE_CARD_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, DEFERRED_UPLOAD_EVENT_LISTENER, or CABLE_CARD_EVENT_LISTENER.

Throws:
- java.lang.SecurityException - if the application does not have MonitorAppPermission("systemevent")

log

public abstract void log(SystemEvent event)

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 Storage API

Package org.ocap.storage

The org.ocap.storage package represents storage devices.

See:

Description

Interface Summary

<table>
<thead>
<tr>
<th>Interface Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AvailableStorageListener</td>
<td>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.</td>
</tr>
<tr>
<td>DetachableStorageOption</td>
<td>This interface represents an external device that can be detached.</td>
</tr>
<tr>
<td>LogicalStorageVolume</td>
<td>This interface represents a logical volume on a storage device.</td>
</tr>
<tr>
<td>RemovableStorageOption</td>
<td>This interface represents a removable storage media bay that supports insertion or removal of storage media while power is applied.</td>
</tr>
<tr>
<td>StorageManagerListener</td>
<td>This interface represents a listener for changes to the set of StorageProxies (StorageProxy).</td>
</tr>
<tr>
<td>StorageOption</td>
<td>This interface represents an option that is specific to a class of storage devices, e.g., detachable or DVR media-capable.</td>
</tr>
<tr>
<td>StorageProxy</td>
<td>This interface represents a persistent storage device.</td>
</tr>
</tbody>
</table>

Class Summary

<table>
<thead>
<tr>
<th>Class Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExtendedFileAccessPermissions</td>
<td>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.</td>
</tr>
<tr>
<td>StorageManager</td>
<td>This class represents the storage manager which keeps track of the storage devices attached to the system.</td>
</tr>
<tr>
<td>StorageManagerEvent</td>
<td>Event sent to a StorageManagerListener registered with the StorageManager that a StorageProxy was added, removed or changed state.</td>
</tr>
</tbody>
</table>

Package org.ocap.storage Description

The org.ocap.storage package represents storage devices.
org.ocap.storage

**Interface AvailableStorageListener**

**All Superinterfaces:**

java.util.EventListener

```java
public interface AvailableStorageListener
    extends java.util.EventListener
```

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

<table>
<thead>
<tr>
<th>Method: notifyHighWaterMarkReached()</th>
</tr>
</thead>
<tbody>
<tr>
<td>notifies the listener a high water mark has been reached in the available memory indicated by <code>dvb.persistent.root</code> and available to all applications.</td>
</tr>
</tbody>
</table>

### Method Detail

**notifyHighWaterMarkReached**

```java
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean isDetachable()</td>
<td>Determines whether the device associated with this storage proxy is ready to be detached.</td>
</tr>
<tr>
<td>void makeDetachable()</td>
<td>Makes the device safe to be detached.</td>
</tr>
<tr>
<td>void makeReady()</td>
<td>Makes the device ready for use.</td>
</tr>
</tbody>
</table>

### Method Detail

**isDetachable**

```java
toolean 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**

```java
void makeDetachable()
```

Throw when the device is unable to make the device safe to detach.

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.
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

class org.ocap.storage.ExtendedFileAccessPermissions
extends FileAccessPermissions

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.

Constructor Summary

ExtendedFileAccessPermissions (boolean readWorldAccessRight,
  boolean writeWorldAccessRight, boolean readOrganisationAccessRight,
  boolean writeOrganisationAccessRight, boolean readApplicationAccessRight,
  boolean writeApplicationAccessRight, int[] otherOrganisationsReadAccessRights,
  int[] otherOrganisationsWriteAccessRights)

  This constructor encodes application, application organisation, and world file access permissions as a set of Booleans, and other organisations file access permissions as arrays of granted organisation identifiers.

Method Summary

int[] getReadAccessOrganizationIds ()
  Gets the array of organisation identifiers with read permission.

int[] getWriteAccessOrganizationIds ()
  Gets the array of organisation identifiers with write permission.

void setPermissions (boolean readWorldAccessRight,
  boolean writeWorldAccessRight, boolean readOrganisationAccessRight,
  boolean writeOrganisationAccessRight, boolean readApplicationAccessRight,
  boolean writeApplicationAccessRight,
  int[] otherOrganisationsReadAccessRights,
  int[] otherOrganisationsWriteAccessRights)

  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 organisation, and world file access permissions as a set of
Booleans, and other organisations file access permissions as arrays of granted organisation identifiers.

Parameters:
readWorldAccessRight - read access for all applications
writeWorldAccessRight - write access for all applications
readOrganisationAccessRight - read access for applications with the same organisation as the
granting application.
writeOrganisationAccessRight - write access for applications with the same organisation as the
granting application.
readApplicationAccessRight - read access for the owner.
writeApplicationAccessRight - write access for the owner.
otherOrganisationsReadAccessRights - array of other organisation identifiers with read access.
Applications with an organisation identifier matching one of these organisation identifiers will be given read
access.
otherOrganisationsWriteAccessRights - array of other organisation identifiers with write
access. Applications with an organisation identifier matching one of these organisation 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[] otherOrganisationsWriteAccessRights)

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 organisation
writeOrganisationAccessRight - write access for organisation
readApplicationAccessRight - read access for the owner
writeApplicationAccessRight - write access for the owner
otherOrganisationsReadAccessRights - array of other organisation identifiers with read access.
Applications with an organisation identifier matching one of these organisation identifiers will be given read
access.
otherOrganizationsWriteAccessRights - array of other organisation identifiers with write access. Applications with an organisation identifier matching one of these organisation identifiers will be given write access.

getReadAccessOrganizationIds
public int[] getReadAccessOrganizationIds()
    Gets the array of organisation identifiers with read permission.
    Returns:
    Array of organisation identifiers with read permission.

getWriteAccessOrganizationIds
public int[] getWriteAccessOrganizationIds()
    Gets the array of organisation identifiers with write permission.
    Returns:
    Array of organisation identifiers with write permission.
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

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>getPath</strong></td>
<td>Gets the absolute path of the volume. This path must be unique across all LogicalStorageVolume instances regardless of the StorageProxy they are contained within.</td>
</tr>
<tr>
<td><strong>getStorageProxy</strong></td>
<td>Gets the StorageProxy the volume is a part of.</td>
</tr>
<tr>
<td><strong>getFileAccessPermissions</strong></td>
<td>Gets the file access permissions of the logical volume.</td>
</tr>
</tbody>
</table>

### 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**

```java
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 removable storage media bay that supports insertion or removal of storage media while power is applied. Examples of removable storage media would be memory stick, CD, or DVD. When an instance of the appropriate storage media is inserted into or removed from the bay a StorageManagerEvent SHALL be generated with an event type of STORAGE_PROXY_CHANGED.

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eject</td>
<td>void eject()</td>
<td>Prepares the storage media to be physically ejected from the bay in an implementation specific fashion, if applicable to the hardware.</td>
</tr>
<tr>
<td>isPresent</td>
<td>boolean isPresent()</td>
<td>Returns a presence indication for the removable storage.</td>
</tr>
</tbody>
</table>

Method Detail

eject

void eject()

Prepares the storage media to be physically ejected from the 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 storage.

Returns:
    True if a removable storage media 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.

Constructor Summary

| protected | StorageManager() |
|-----------|
| Protected default constructor. |

Method Summary

| 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. |

| abstract void | addStorageManagerListener |
| (StorageManagerListener listener) |
| Adds a listener to receive StorageManagerEvents when a storage proxy is added, removed or changes state. |

| 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. |

| static StorageManager | getInstance() |
| Gets the singleton instance of the storage manager. |

| abstract StorageProxy[] | getStorageProxies() |
| Gets the set of StorageProxy instances representing all of the currently attached or embedded storage devices. |

| 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. |

| abstract void | removeAvailableStorageListener |
| (AvailableStorageListener listener) |
| Removes an available storage listener that was registered using the addAvailableStorageListener method. |

| abstract void | removeStorageManagerListener |
| (StorageManagerListener listener) |
| Removes a listener so that it no longer receives StorageManagerEvents when storage proxies change. |
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

StorageManager
protected StorageManager()
Protected default constructor.

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)
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 is null.

removeStorageManagerListener
public abstract void removeStorageManagerListener(StorageManagerListener listener)
Removes a listener so that it no longer receives StorageManagerEvents when storage proxies change. This method has no effect if the given listener had not been added.
Parameters:
listener - The storage manager listener to be removed.
Throws:
java.lang.IllegalArgumentException - if the listener parameter is null.
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.

See Also:
   Serialized Form

Field Summary

static int STORAGE_PROXY_ADDED
   A StorageProxy was added.

static int STORAGE_PROXY_CHANGED
   A StorageProxy changed state.

static int STORAGE_PROXY_REMOVED
   A StorageProxy was removed.

Fields inherited from class java.util.EventObject

source

Constructor Summary

StorageManagerEvent(StorageProxy proxy, int eventType)
   Constructs the event.

Method Summary

int getEventType()
   Returns the event type.

StorageProxy getStorageProxy()
   Returns the StorageProxy that caused the event.

Methods inherited from class java.util.EventObject

ggetSource, 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.
See Also:
Constant Field Values

STORAGE_PROXY_REMOVED
public static final int STORAGE_PROXY_REMOVED
A StorageProxy was removed.
See Also:
Constant Field Values

STORAGE_PROXY_CHANGED
public static final int STORAGE_PROXY_CHANGED
A StorageProxy changed state.
See Also:
Constant Field Values

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.
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.

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void notifyChange(StorageManagerEvent sme)</td>
</tr>
<tr>
<td>The implementation calls this method to inform the listener when a storage proxy was added, removed or changes state.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>notifyChange</td>
</tr>
<tr>
<td>void notifyChange(StorageManagerEvent sme)</td>
</tr>
<tr>
<td>The implementation calls this method to inform the listener when a storage proxy was added, removed or changes state. changed.</td>
</tr>
<tr>
<td>Parameters:</td>
</tr>
<tr>
<td>sme - Event indicating the change.</td>
</tr>
<tr>
<td>See Also:</td>
</tr>
<tr>
<td>StorageManagerEvent</td>
</tr>
</tbody>
</table>
org.ocap.storage

**Interface StorageOption**

**All Known Subinterfaces:**
DetachableStorageOption, MediaStorageOption, RemovableStorageOption

```java
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:
   ResourceProxy

public interface StorageProxy
extends 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 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 int | 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. |
| static int | DEVICE_ERROR | Returned by getStatus() to indicate that the device is in an unrecoverable error state and cannot be used. |
| static int | NOT_PRESENT | Returned by getStatus() to indicate that a detected storage device bay does not contain a removable storage device, i.e. |
| static int | 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()). |
| static int | READY | Returned by getStatus() to indicate that the device is initialized, mounted and ready for use. |
| static int | UNINITIALIZED | Returned by getStatus() to indicate that the device is completely uninitialized and contains no existing data. |
Field Summary

static int UNSUPPORTED_DEVICE
  Returned by getStatus() to indicate that the device that has been plugged in is not supported by the platform.

static int 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.

Method Summary

LogicalStorageVolume allocateGeneralPurposeVolume(java.lang.String name, ExtendedFileAccessPermissions fap)
  Allocates a general purpose LogicalStorageVolume.

void deleteVolume(LogicalStorageVolume vsp)
  Deletes a LogicalStorageVolume.

java.lang.String getDisplayName()
  Gets a storage device name that can be displayed to a user for selection.

long getFreeSpace()
  Gets the available storage capacity in bytes.

java.lang.String getName()
  Gets the storage device name assigned by the implementation.

StorageOption[] getOptions()
  Gets the array of storage device options (e.g., DetachableStorageOption).

int getStatus()
  Returns the status of the storage device.

boolean[] getSupportedAccessRights()
  Gets the permissions supported by this storage device.

long getTotalSpace()
  Gets the total storage capacity of the device in bytes.

LogicalStorageVolume[] getVolumes()
  Gets the set of logical volumes see present on the StorageProxy.

void initialize(boolean userAuthorized)
  Initializes the StorageProxy for use.

Methods inherited from interface org.davic.resources.ResourceProxy

getClient

Field Detail

READY

static final int READY
  Returned by getStatus() to indicate that the device is initialized, mounted and ready for use.
See Also:
Constant Field Values

OFFLINE

\texttt{static final int OFFLINE}

Returned by \texttt{getStatus()} to indicate that the device is present but some other action is required before the device can be used (e.g., \texttt{DetachableStorageOption.makeReady()}).

See Also:
Constant Field Values

BUSY

\texttt{static final int BUSY}

Returned by \texttt{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.

See Also:
Constant Field Values

UNSUPPORTED_DEVICE

\texttt{static final int UNSUPPORTED_DEVICE}

Returned by \texttt{getStatus()} to indicate that the device that has been plugged in is not supported by the platform.

See Also:
Constant Field Values

UNSUPPORTED_FORMAT

\texttt{static final int UNSUPPORTED_FORMAT}

Returned by \texttt{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.

See Also:
Constant Field Values

UNINITIALIZED

\texttt{static final int UNINITIALIZED}

Returned by \texttt{getStatus()} to indicate that the device is completely uninitialized and contains no existing data. It must be initialized by calling the \texttt{initialize} method to make the device is usable.

See Also:
Constant Field Values

DEVICE_ERROR

\texttt{static final int DEVICE_ERROR}

Returned by \texttt{getStatus()} to indicate that the device is in an unrecoverable error state and cannot be used.

See Also:
Constant Field Values
NOT_PRESENT

static final int 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.
    See Also:
    Constant Field Values

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

int 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

```java
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

```java
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

```java
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)`).

The name parameter SHALL be used as the last directory name in the absolute path of the logical storage volume when created.

**Parameters:**
- **name**: Name of the new LogicalStorageVolume. Must be unique for the organization and application identifiers in the path on this StorageProxy.
- **fap**: Application access permissions of the new LogicalStorageVolume. Applies to the last directory in the path returned by getPath.

**Returns:**
Allocated volume storage proxy.

**Throws:**
- `java.lang.IllegalArgumentException` - if the name does not meet the persistentfilename form specified by DVB-MHP 1.0.3 chapter 14, or if the length is greater than 255, 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. The `getSupportedAccessRights` method indicates if the StorageProxy can be written to by the calling application.
- `java.lang.SecurityException` - if the calling application does not have persistent storage permission as requested by its permission request file.
deleteVolume

```java
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. This method deletes the volume regardless of file locks.

**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

```java
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 Test API

Package org.ocap.test
OCAP testing communications.

Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCAPTest</td>
<td>The purpose of this class is to provide a very simple communication channel between the OCAP implementation under test (IUT) and the test server.</td>
</tr>
</tbody>
</table>

Package org.ocap.test Description

OCAP testing communications

OCAP testing requires the test applications executing on OCAP devices be able to communicate with an Automated Test Environment (ATE) in a manufacturer-independent manner. 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 indicates that the OCAP devices SHALL communicate with the ATE using a common, defined communications mechanism.

Since OCAP devices, by definition, must support TCP/IP and UDP/IP for interaction channel protocols, TCP/IP and UDP/IP sockets have been chosen as the mechanism by which OCAP devices will communicate with an OCAP ATE. The interaction channel provides either TCP/IP or UDP/IP protocol, via the RF connector labeled 'Cable In' on the OCAP Host Device, and the CableCARD. Though the OCAP Host Device supports both TCP/IP and UDP/IP, only one of them SHALL be used for the ATE, according to the type of CableCARD, once communication between the ATE and Host device has been established (see item 'b)' below).

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 W–1.
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 transmit a location of 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 multiplexed 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 the 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):

a) 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.

b) The ATE begins sending IP datagrams to the IP address of the OCAP Host Device it is associated with.

c) The OCAP Host Device powers up out of reset and reads a 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.

d) The Test Monitor Application invokes \texttt{OcapSystem.monitorConfiguredSignal()} to enable conformance testing APIs (\texttt{org.ocap.test.OCAPTest}). When \texttt{monitorConfiguredSignal} is called, the implementation will enable the conformance testing APIs (see \texttt{org.ocap.OcapSystem.monitorConfiguredSignal} for details and OCAP testing communications below).

e) Test Monitor Application requests configuration parameters including information on the initial channel to tune to using the communications channel established in step c).

f) Tuning to the specified channel initiates a service containing the bound testlet application.

g) Test application executes on OCAP Host Device, communicating with the ATE as required via the communications channel opened up in step 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 manner. The need to communicate with an ATE in a manufacturer-independent manner means that OCAP Host Devices SHALL communicate with the ATE using a common, bi-directional interaction channel. In an effort to isolate OCAP Host Device implementations, from the need to implement interaction channel protocols, which are necessary only for use in compliance testing, the class \texttt{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:
Interaction subsystem Design

![Interaction Subsystem Design Diagram](image)

Though a headend system is implemented to follow some standard specs to support the standard CableCARD, it has some restrictions on its software and hardware. For example, even though an OCAP implementation must support both TCP/IP and UDP/IP on the OOB return channel, the headend may not support TCP/IP that supports the stable 'ack' protocol. For such a case, the OCAP automatic test environment emulates the 'ack' protocol for UDP/IP to ensure the return channel communication to collect the test result with certainty. However, the headend restriction may change in the future. The OCAPTest class provides simple communication methods so the OCAP ATE can implement a specific overlay protocol to avoid the headend restrictions.

The OCAPTest provides two methods to send a raw message data. The sendUDP() sends the rawMessage by a single UDP packet. OCAPTest doesn't care what is written in the rawMessage. The terminal byte is handled by the test application and ATE automatically. Contrarily, the send() sends the rawMessage by TCP and needs to send a terminal byte. The OCAP ATE specifies the format of rawMessage and analyzes it by its own parser, such that the OCAP implementation is unknowing of the rawMessage format and simplifies any future update to the OCAP ATE protocol.

Two protocols; UDP and TCP are distinguished by a message from ATE. The ATE sends an "ate:a.b.c.d:xxxx:ppp" message to a host device in an initial phase of testing. See the description of the org.ocap.OcapSystem class.
If a ppp value in an "ate:a.b.c.d:xxxx:ppp" message from ATE is "TCP", an OCAPTest instance shall be configured to use TCP/IP, i.e., the send(String) and receive() methods are used by a test application. TCP/IP protocol supports a trusted communication using TCP acknowledge packet.

Contrarily, if the ppp value is "UDP", an OCAPTest instance shall be configured to use UDP/IP, (i.e., the sendUDP(byte) and receiveUDP() methods are used by a test application). Since UDP/IP doesn't support an acknowledge packet, the test application emulates the trusted acknowledge communication like TCP/IP. However, the acknowledge protocol is implemented in the ATE protocol so that the OCAPTest doesn't have to implement anything about the acknowledge protocol. Though the UDP communication may have a restriction of the payload length according to a current physical layer, the OCAPTest implementation shall not divide the payload since the test application will divide the original message into a proper length for the physical layer, then call the sendUDP() method.

The test application will distinguish the protocol to be used with an OCAPTest.getProtocol() method. The getProtocol() must return the protocol type specified by an "ate:a.b.c.d:xxxx:ppp" message.

Testing Scenario:
   a) The ATE's test supervisor is downloaded on the OCAP host;
   b) The supervisor calls the OcapSystem.monitorConfiguration() method, and then calls OCAPTest.getProtocol() method;
   c) Then the supervisor can use either sendUDP() or send() according to the return value of the getProtocol() method.

NOTE: The actual test case code may not have to know the protocol type since the supervisor may have an overlay method for easy message handling. It depends on the version of ATE.

Example Code

The following is an example implementation of an OCAPTest. It is supplied as a reference for your convenience only, and was used to test the OCAPTest APIs. This code is provided "as is", without support or maintenance, and without any warranty or representation of any kind, including, but not limited to, any implied warranty of fitness for a particular use, or non-infringement. The use of the code shall be the responsibility of the implementer.

NOTE: CableLabs disclaims any and all responsibility for implementer's use of the following code.

/*
 * This is sample implementation of OCAPTest.
 */

package org.ocap.test;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.DatagramSocket;
import java.net.DatagramPacket;
import java.net.InetAddress;
import java.security.AccessController;
import java.security.PrivilegedExceptionAction;
import java.security.PrivilegedActionException;

public class OCAPTest {

    /**
     * Message termination byte.
     */
    public static final byte MESSAGE_TERMINATION_BYTE = '\0';

    /**
     * Maximum length of messages exchanged between ATE and a test application.
     */
    public static final int MAX_MESSAGE_LENGTH = 1500;

    /**
     * Indicates that an OCAPTest instance is configured to use UDP/IP
     * protocol.
     */
    public static final int UDP = 0;

    /**
     * Indicates that an OCAPTest instance is configured to use TCP/IP
     * protocol.
     */
    public static final int TCP = 1;

    private static int protocol = UDP;
    private static Socket tcpSocket = null;
    private static InetAddress serverAddr = null;
    private static DataInputStream dataIn = null;
    private static DataOutputStream dataOut = null;
    private static DatagramSocket udpSocket = null;

    /*
     * Initialize sockets. This method is called only in
     * OcapSystem.monitorConfiguredSignal(port, timeout) with privileges
     * enabled.
     *
     * @param serverName the IP address of ATE used for OCAP testing, that is
     * specified a a.b.c.d value in an "ate:a.b.c.d:xxxx:ppp" message from
     * ATE.
     * @param interactionPort the IP port number used for OCAP testing, that
     * is specified a xxxx value in an "ate:a.b.c.d:xxxx:ppp" message from
     * ATE.
     * @param interactionProtocol the protocol used for OCAP testing, that
     * is specified a ppp value in an "ate:a.b.c.d:xxxx:ppp" message from
     * ATE.
     * @param myName the IP address of IUT used for OCAP testing, that will be
     * provided by a CableCARD.
     */
}
static void initialize(
    String serverName, int interactionPort, int interactionProtocol,
    String myName) throws IOException {
    serverAddr = InetAddress.getByName(serverName);
    if (interactionProtocol == TCP) {
        tcpSocket = new Socket(serverAddr, interactionPort);
        dataIn = new DataInputStream(new
            BufferedInputStream(tcpSocket.getInputStream()));
        dataOut = new DataOutputStream(new
            BufferedOutputStream(tcpSocket.getOutputStream()));
    } else {
        InetAddress myAddress = InetAddress.getByName(myName);
        udpSocket = new DatagramSocket(interactionPort, myAddress);
    }
    protocol = interactionProtocol;
}

public static byte[] receive() throws IOException {
    byte[] recvBuf = new byte[MAX_MESSAGE_LENGTH];
    int len = 0;

    while (true) {
        recvBuf[len] = (byte) (dataIn.read() & 0xff);
        if (recvBuf[len] == MESSAGE_TERMINATION_BYTE) {
            break;
        }
        len++;
    }
    byte[] message = new byte[len];
    System.arraycopy(recvBuf, 0, message, 0, len);
    return message;
}

public static void send(byte[] rawMessage) throws IOException {
    for (int i = 0; i < rawMessage.length; i++) {
        if (rawMessage[i] == MESSAGE_TERMINATION_BYTE) {
            throw new IllegalArgumentException("rawMessage contains a "+
                "byte with the value MESSAGE_TERMINATION_BYTE.");
        }
    }
}
dataOut.write(rawMessage, 0, rawMessage.length);
dataOut.WriteByte(MESSAGE_TERMINATION_BYTE);
dataOut.flush();
}

/**
 * Returns a current protocol that is used by an OCAPTest instance.
 */
public static int getProtocol() {
    return protocol;
}

/**
 * Receive a message from ATE via UDP/IP.
 */
public static byte[] receiveUDP() throws IOException {
    byte[] datas = new byte[MAX_MESSAGE_LENGTH];
    final DatagramPacket dp = new DatagramPacket(datas, datas.length);
    try {
        AccessController.doPrivileged(
            new PrivilegedExceptionAction() {
                public Object run() throws Exception {
                    udpSocket.receive(dp);
                    return null;
                }
            });
    } catch (PrivilegedActionException pae) {
        Exception ex = pae.getException();
        if (ex instanceof IOException) {
            throw (IOException) ex;
        } else if (ex instanceof RuntimeException) {
            throw (RuntimeException) ex;
        } else {
            throw new InternalError("Bug of receiveUDP: " + ex);
        }
    }
    byte[] realDatas = new byte[dp.getLength()];
    System.arraycopy(datas, 0, realDatas, 0, dp.getLength());
    return realDatas;
}
/**
 * Send rawMessage to ATE via UDP/IP.
 */
public static void sendUDP(byte[] rawMessage) throws java.io.IOException {
    int interactionPort = udpSocket.getLocalPort();
    final DatagramPacket dp = new DatagramPacket(
            rawMessage, rawMessage.length, serverAddr, interactionPort);

    try {
        AccessController.doPrivileged(
            new PrivilegedExceptionAction() {
                public Object run() throws Exception {
                    udpSocket.send(dp);
                    return null;
                }
            });
    } catch (PrivilegedActionException pae) {
        Exception ex = pae.getException();
        if (ex instanceof IOException) {
            throw (IOException) ex;
        } else if (ex instanceof RuntimeException) {
            throw (RuntimeException) ex;
        } else {
            throw new InternalError("Bug of sendUDP: " + ex);
        }
    }
}
org.ocap.test
Class OCAPTest

class 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

<table>
<thead>
<tr>
<th>static int</th>
<th>MAX_MESSAGE_LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Maximum length of messages exchanged between ATE and a test application.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static byte</th>
<th>MESSAGE_TERMINATION_BYTE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Message termination byte.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Indicates that an OCAPTest instance is configured to use TCP/IP protocol.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Indicates that an OCAPTest instance is configured to use UDP/IP protocol.</td>
</tr>
</tbody>
</table>

Constructor Summary

OCAPTest()
**Method Summary**

<table>
<thead>
<tr>
<th>static void</th>
<th>send(byte[] rawMessage)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Send a specified byte array to ATE via TCP/IP protocol.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static void</th>
<th>sendUDP(byte[] rawMessage)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Send rawMessage to ATE via UDP/IP.</td>
</tr>
</tbody>
</table>

**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.

See Also:
Constant Field Values

**MAX_MESSAGE_LENGTH**

public static final int MAX_MESSAGE_LENGTH

Maximum length of messages exchanged between ATE and a test application.

See Also:
Constant Field Values

**UDP**

public static final int UDP

Indicates that an OCAPTest instance is configured to use UDP/IP protocol.

See Also:
Constant Field Values

**TCP**

public static final int TCP

Indicates that an OCAPTest instance is configured to use TCP/IP protocol.

See Also:
Constant Field Values

**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  (reserved)
Annex Y  Environment API

Package org.ocap.environment

Represents an environment that provides the context in which applications run.

See:

Description

## Interface Summary

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnvironmentListener</td>
<td>The listener interface for receiving environment events.</td>
</tr>
</tbody>
</table>

## Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment</td>
<td>Represents an environment that provides the context in which applications run.</td>
</tr>
<tr>
<td>EnvironmentEvent</td>
<td>The EnvironmentEvent class is used to notify applications of events relating to environments.</td>
</tr>
<tr>
<td>EnvironmentState</td>
<td>Defines the set of available states for an environment.</td>
</tr>
<tr>
<td>EnvironmentStateChangedEvent</td>
<td>The EnvironmentStateChangedEvent class indicates the completion of a state transition of an environment.</td>
</tr>
</tbody>
</table>

Package org.ocap.environment Description

Represents an environment that provides the context in which applications run.
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.

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.
- Environments in the background state may have running applications but these applications SHALL NOT be in the normal mode.

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 pauseable 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: Applicable events announcing such changes are generated as expected. For example, org.dvb.application.AppStateChangeEvents announcing that applications have been paused or terminated; and java.awt.event.ComponentEvents announcing that an HScene has been hidden.

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 if their service is still selected
• 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 policy of the newly selected environment is responsible for determining which of the applications in that environment should be the first to have focus.

• Any pauseable 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

<table>
<thead>
<tr>
<th>protected</th>
<th>Constructor for environments.</th>
</tr>
</thead>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addEnvironmentListener(EventListener l)</td>
<td>Add a listener for environment events.</td>
</tr>
<tr>
<td>void deselect()</td>
<td>Request this environment cease being selected.</td>
</tr>
<tr>
<td>static Environment getHome()</td>
<td>Return the calling applications home environment</td>
</tr>
<tr>
<td>EnvironmentState getState()</td>
<td>Queries the state of this environment.</td>
</tr>
<tr>
<td>void removeEnvironmentListener(EventListener l)</td>
<td>Remove a listener for environment events.</td>
</tr>
<tr>
<td>void select()</td>
<td>Request this environment become selected.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Constructor Detail

**Environment**

<table>
<thead>
<tr>
<th>protected</th>
<th>Constructor for environments. This is provided for the use of implementations or other specifications and is not to be used by applications.</th>
</tr>
</thead>
</table>
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(IllegalArgumentException - 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.

See Also:
   Serialized Form

Field Summary

Fields inherited from class java.util.EventObject
source

Constructor Summary

EnvironmentEvent(Environment source)
   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

All Superinterfaces:
java.util.EventListener

public interface EnvironmentListener
extends java.util.EventListener

The listener interface for receiving environment events.

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void notify(EnvironmentEvent e)</td>
</tr>
<tr>
<td>Invoked when an application is to be notified of an event relating to an environment</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>notify</td>
</tr>
<tr>
<td>void notify(EnvironmentEvent e)</td>
</tr>
<tr>
<td>Invoked when an application is to be notified of an event relating to an environment</td>
</tr>
<tr>
<td>Parameters:</td>
</tr>
<tr>
<td>e - the event</td>
</tr>
</tbody>
</table>
**org.ocap.environment**

**Class EnvironmentState**

```java
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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static EnvironmentState <strong>BACKGROUND</strong></td>
<td>The environment is in the background.</td>
</tr>
<tr>
<td>static EnvironmentState <strong>INACTIVE</strong></td>
<td>The environment is inactive.</td>
</tr>
<tr>
<td>static EnvironmentState <strong>PRESENTING</strong></td>
<td>The environment is presenting.</td>
</tr>
<tr>
<td>static EnvironmentState <strong>SELECTED</strong></td>
<td>The environment is selected.</td>
</tr>
</tbody>
</table>

### Constructor Summary

```java
protected EnvironmentState(java.lang.String s)
```

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**

```java
public static final EnvironmentState INACTIVE
```

The environment is inactive. No applications are running.

**SELECTED**

```java
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 may be visible to the end user. They cannot receive user input from the remote control.

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

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 previously terminated applications have been re-started as described as part of the transition to the selected state (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 pauseable 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.

See Also:
- Serialized Form
Field Summary

Fields inherited from class java.util.EventObject

source

Constructor Summary

EnvironmentStateChangedEvent(Environment source, EnvironmentState fromstate, EnvironmentState tostate)

Create an EnvironmentStateChangedEvent object.

Method Summary

EnvironmentState getFromState()

Return the state the environment was in before the state transition was requested as passed to the constructor of this event.

EnvironmentState getToState()

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(Environmst 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

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.
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
## Annex Z  Diagnostics API

### Package org.ocap.diagnostics

<table>
<thead>
<tr>
<th><strong>Interface Summary</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MIBDefinition</strong></td>
<td>This interface represents a MIB object that exposes its data type.</td>
</tr>
<tr>
<td><strong>MIBListener</strong></td>
<td>This interface represents a listener that can be registered with the MIBManager in order to listen for requests to MIB object encodings.</td>
</tr>
<tr>
<td><strong>SNMPRequest</strong></td>
<td>This interface represents an application request for SNMP check, get, or set of a specific MIB.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Class Summary</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MIBManager</strong></td>
<td>The MIBManager class provides Management Information Base (MIB) access to the host's MIB.</td>
</tr>
<tr>
<td><strong>MIBObject</strong></td>
<td>This interface represents a MIB Object.</td>
</tr>
<tr>
<td><strong>SNMPResponse</strong></td>
<td>This interface represents a response to an implementation request to an application that has registered control over a specific MIB.</td>
</tr>
</tbody>
</table>
org.ocap.diagnostics

Interface MIBDefinition

public interface MIBDefinition

This interface represents a MIB object that exposes its data type. See RFC 2578 for data type definition.

### Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int SNMP_TYPE_BITS</td>
<td>The BITS construct.</td>
</tr>
<tr>
<td>static int SNMP_TYPE_COUNTER32</td>
<td>Base type, application defined 32 bit counter.</td>
</tr>
<tr>
<td>static int SNMP_TYPE_COUNTER64</td>
<td>Base type, application defined 64 bit counter.</td>
</tr>
<tr>
<td>static int SNMP_TYPE_GAUGE32</td>
<td>Base type, application defined 32 bit gauge.</td>
</tr>
<tr>
<td>static int SNMP_TYPE_INTEGER</td>
<td>Base type, built-in ASN.1 integer type.</td>
</tr>
<tr>
<td>static int SNMP_TYPE_INVALID</td>
<td>Unrecognized type encountered.</td>
</tr>
<tr>
<td>static int SNMP_TYPE_IPADDRESS</td>
<td>Base type, application defined IP address.</td>
</tr>
<tr>
<td>static int SNMP_TYPE_OBJECTID</td>
<td>Base type, built-in ASN.1 OBJECT IDENTIFIER type.</td>
</tr>
<tr>
<td>static int SNMP_TYPE_OCTETSTRING</td>
<td>Base type, built-in ASN.1 string type.</td>
</tr>
<tr>
<td>static int SNMP_TYPE_OPAQUE</td>
<td>Base type, application defined opaque variable.</td>
</tr>
<tr>
<td>static int SNMP_TYPE_TIMETICKS</td>
<td>Base type, application defined time ticks.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int getDataType()</td>
<td>Gets the SNMP data type of the MIB.</td>
</tr>
<tr>
<td>MIBObject getMIBObject()</td>
<td>Gets the MIB object associated with this MIB definition.</td>
</tr>
</tbody>
</table>

### Field Detail

SNMP_TYPE_INVALID

static final int SNMP_TYPE_INVALID
Unrecognized type encountered. Not defined by RFC 2578.

See Also:
Constant Field Values

SNMP_TYPE_INTEGER

static final int SNMP_TYPE_INTEGER
Base type, built-in ASN.1 integer type.
See Also:
Constant Field Values

SNMP_TYPE_BITS

static final int SNMP_TYPE_BITS
The BITS construct.
See Also:
Constant Field Values

SNMP_TYPE_OCTETSTRING

static final int SNMP_TYPE_OCTETSTRING
Base type, built-in ASN.1 string type.
See Also:
Constant Field Values

SNMP_TYPE_OBJECTID

static final int SNMP_TYPE_OBJECTID
Base type, built-in ASN.1 OBJECT IDENTIFIER type.
See Also:
Constant Field Values

SNMP_TYPE_IPADDRESS

static final int SNMP_TYPE_IPADDRESS
Base type, application defined IP address.
See Also:
Constant Field Values

SNMP_TYPE_COUNTER32

static final int SNMP_TYPE_COUNTER32
Base type, application defined 32 bit counter.
See Also:
Constant Field Values

SNMP_TYPE_GAUGE32

static final int SNMP_TYPE_GAUGE32
Base type, application defined 32 bit gauge.
See Also:
Constant Field Values
SNMP_TYPE_TIMETICKS
static final int SNMP_TYPE_TIMETICKS
   Base type, application defined time ticks.
See Also:
   Constant Field Values

SNMP_TYPE_OPAQUE
static final int SNMP_TYPE_OPAQUE
   Base type, application defined opaque variable.
See Also:
   Constant Field Values

SNMP_TYPE_COUNTER64
static final int SNMP_TYPE_COUNTER64
   Base type, application defined 64 bit counter.
See Also:
   Constant Field Values

Method Detail

getDataType
int getDataType()
   Gets the SNMP data type of the MIB.
Returns:
   An SNMP data type defined by constants in this interface.

getMIBObject
MIBObject getMIBObject()
   Gets the MIB object associated with this MIB definition.
Returns:
   The MIB Object for this MIB definition.
org.ocap.diagnostics

**Interface MIBListener**

All Superinterfaces:
java.util.EventListener

```java
public interface MIBListener
extends java.util.EventListener
```

This interface represents a listener that can be registered with the MIBManager in order to listen for requests to MIB object encodings.

### Method Summary

<table>
<thead>
<tr>
<th>SNMPResponse</th>
<th>notifySNMPRequest(SNMPRequest request)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Notifies a listener when a MIB object it registered control over has been requested.</td>
</tr>
</tbody>
</table>

### Method Detail

**notifySNMPRequest**

```java
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.
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.

### Field Summary

| static int | ECM_SUBDEVICE | Used to identify the subdevice type and query eCM MIB values. |
| static int | ESTB_SUBDEVICE | Used to identify the subdevice type and query Host MIB values. |
| static int | MIB_ACCESS_READONLY |
| static int | MIB_ACCESS_READWRITE |
| static int | MIB_ACCESS_WRITEONLY |

### Constructor Summary

| protected | MIBManager() | Protected constructor, no application access. |

### Method Summary

| static MIBManager | getInstance() | Gets the MIBManager. |
| abstract MIBDefinition[] | queryMibs(int subDevice, java.lang.String oid) | Makes a query for all MIB objects matching the oid parameter, as well as any descendants in the MIB tree, in a specific sub-device including eSTB and eCM. |
| 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. |
| abstract void | registerOID(java.lang.String oid, int access, boolean leaf, int dataType, MIBListener listener) | Registers a MIB object by adding the OID and listener to the MIB tables. |
Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract void setMIBObject(MIBObject mibToSet)</td>
<td>Sets a Host MIB object.</td>
</tr>
<tr>
<td>abstract void unregisterOID(java.lang.String oid)</td>
<td>Unregisters a previously registered OID if the OID was registered by the same application.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

MIB_ACCESS_READONLY

public static final int MIB_ACCESS_READONLY

See Also:
Constant Field Values

MIB_ACCESS_READWRITE

public static final int MIB_ACCESS_READWRITE

See Also:
Constant Field Values

MIB_ACCESS_WRITEONLY

public static final int MIB_ACCESS_WRITEONLY

See Also:
Constant Field Values

ECM_SUBDEVICE

public static final int ECM_SUBDEVICE

Used to identify the subdevice type and query eCM MIB values.

See Also:
Constant Field Values

ESTB_SUBDEVICE

public static final int ESTB_SUBDEVICE

Used to identify the subdevice type and query Host MIB values.

See Also:
Constant Field Values

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, boolean leaf, 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 include a trailing ".0" for leaf items or but not include the instance ID for table column items. (The same listener may be installed to will be invoked for all row instances for table column items. 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.
leaf - When true indicates if the object is a leaf, otherwise the object is a non-leaf table column.
dataType - The data type of the MIB being registered. See constants in MIBDefinition for valid values.
listener - Listener to the MIB being registered. Must not 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. If the object to be searched for is a leaf the trailing ".0" must be included for an exact match. A query for a leaf object SHALL return just that object if found. A query for a non-leaf OID SHALL return all MIB objects below that OID. Existing leaf and table items SHALL be included in the results; branch-nodes without data SHALL NOT. For example; If a query is for OID 1.2.3.4 then all table items and leaves below that OID are returned. If OIDs 1.2.3.4.1 and 1.2.3.4.2 are the only items below the query object they would be returned. The query SHALL NOT return items outside the OID. For example; if 1.2.3.4 is the query OID then 1.2.3.5 is not returned.

If the CableCARD supports MIB APDUs then this method SHALL be able to query the CableCARD MIB objects.

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 lexographically ordered by increasing value of OID with the lowest value in the first element of the array.

queryMibs

public abstract MIBDefinition[] queryMibs(int subDevice,
java.lang.String oid)

Makes a query for all MIB objects matching the oid parameter, as well as any descendants in the MIB tree, in a specific sub-device including eSTB and eCM. If the object to be searched for is a leaf the trailing ".0" must be included for an exact match. A query for a leaf object SHALL return just that object if found. A query for a non-leaf OID SHALL return all MIB objects below that OID. Existing leaf and table items SHALL be included in the results; branch-nodes without data SHALL NOT. For example; If a query is for OID 1.2.3.4 then all table items and leaves below that OID are returned. If OIDs 1.2.3.4.1 and 1.2.3.4.2 are the only items below the query object they would be returned. The query SHALL NOT return items outside the OID. For example; if 1.2.3.4 is the query OID then 1.2.3.5 is not returned.

If ECM_SUBDEVICE is the subDevice argument value the implementation SHALL allow eCM MIB query in an implementation specific fashion. The Host device may restrict eCM MIB access to a particular subset of MIB objects; see Host CFR for required eCM MIB support.

Parameters:
subDevice - The sub-device to read the MIB from. Using ESTB_SUBDEVICE is equivalent to calling queryMibs(String).
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 lexographically ordered by increasing value of OID with the lowest value in the first element of the array.

Throws:
java.lang.IllegalArgumentException - if the subDevice is not one of the ECM_SUBDEVICE or ESTB_SUBDEVICE.

setMIBObject

public abstract void setMIBObject(MIBObject mibToSet)

Sets a Host MIB object. The OID in the MIBObject parameter must reference a writable MIB object supported by the implementation.

Note; This method can only set writable MIBs in the Host device (i.e. ESTB_SUBDEVICE). Writable MIBs for other sub-devices such as CableCARD and eCM cannot be set using this method.

Parameters:
mibToSet - The MIB to set. The OID in the MIBObject SHALL be used to replace values with the data
array in the parameter.

Throws:
java.lang.IllegalArgumentException - if the OID in the MIBObject parameter does not
reference a supported Host MIB or if the data attribute in the MIBObject parameter is not in the correct
ASN.1 form for the OID.
java.lang.SecurityException - if the MIB is not writable or if the calling application does not
have MonitorAppPermission("diagnostics").
openCable Application Platform (OCAP)
OC-SP-OCAP1.3.1-130530

**org.ocap.diagnostics**

**Class MIBObject**

```java
public class MIBObject extends java.lang.Object {
    // Constructor
    public MIBObject(java.lang.String oid, byte[] data) {
        // Constructor body
    }

    // Method
    public java.lang.String getOID() {
        // Method body
    }

    // Method
    public byte[] getValue() {
        // Method body
    }
}
```

This interface represents a MIB Object.

**Constructor Summary**

`MIBObject(java.lang.String oid, byte[] data)`

Constructs a MIB object.

**Method Summary**

- `byte[] getData()`
  - Returns the current MIB object data, in ASN.1 format with BER encoding.
- `java.lang.String getOID()`
  - Gets the MIB object identifier.
- `byte[] getValue()`
  - Returns the unencoded value of the current MIB object data.

**Methods inherited from class java.lang.Object**

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

**Constructor Detail**

`MIBObject(java.lang.String oid, byte[] data)`

Constructs a MIB object.

**Parameters:**
- `oid` - Object Identifier of the MIB object.
- `data` - Array of bytes representing the MIB object data, in ASN.1 format with BER encoding.

**Method Detail**

`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**

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

Returns the current MIB object data, in ASN.1 format with BER encoding.

**Returns:**
A byte array containing the ASN.1 formatted MIB object data.

**getValue**

```java
public byte[] getValue()
```

Returns the unencoded value of the current MIB object data. The returned array is the value field of the BER type-length-value encoding provided by `getData()`.

**Returns:**
A byte array representing the unencoded value of the MIB object data, or `null` if the data cannot be unencoded.

**See Also:**
`getData()`
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

<table>
<thead>
<tr>
<th>Static int</th>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SNMP_CHECK_FOR_SET_REQUEST</td>
<td>Used to validate a MIB value before doing a set.</td>
</tr>
<tr>
<td></td>
<td>SNMP_GET_NEXT_REQUEST</td>
<td>Get data for the next OID beyond the one passed in.</td>
</tr>
<tr>
<td></td>
<td>SNMP_GET_REQUEST</td>
<td>Get data for the exact OID passed in.</td>
</tr>
<tr>
<td></td>
<td>SNMP_SET_REQUEST</td>
<td>Set (modify) the value in the MIB for an OID.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIBObject</td>
<td>getMIBObject()</td>
<td>Gets the MIBObject for the request.</td>
</tr>
<tr>
<td>int</td>
<td>getType()</td>
<td>Gets the type for this request.</td>
</tr>
</tbody>
</table>

### Field Detail

**SNMP_CHECK_FOR_SET_REQUEST**

static final int SNMP_CHECK_FOR_SET_REQUEST

Used to validate a MIB value before doing a set.

See Also:

Constant Field Values

**SNMP_SET_REQUEST**

static final int SNMP_SET_REQUEST

Set (modify) the value in the MIB for an OID.

See Also:

Constant Field Values

**SNMP_GET_REQUEST**

static final int SNMP_GET_REQUEST

Get data for the exact OID passed in.

See Also:

Constant Field Values
SNMP_GET_NEXT_REQUEST

static final int SNMP_GET_NEXT_REQUEST
    Get data for the next OID beyond the one passed in.

See Also:
Constant Field Values

Method Detail

getRequestType

int getRequestType()
    Gets the type for this request.
    Returns:
    One of the request types defined in this interface.

getMIBObject

MIBObject getMIBObject()
    Gets the MIBObject for the request.
    Returns:
    MIBObject for the request.
org.ocap.diagnostics
Class SNMPResponse

java.lang.Object
   org.ocap.diagnostics.SNMPResponse

public class SNMPResponse
extends java.lang.Object

This interface represents a response to an implementation request to an application that has registered control over a specific MIB.

Field Summary

<table>
<thead>
<tr>
<th>static int</th>
<th>SNMP_REQUEST_AUTHORIZATION_ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A problem occurred in authorization.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>SNMP_REQUEST_BAD_VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A Check/Set value (or syntax) error occurred.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>SNMP_REQUEST_COMMIT_FAILED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An attempt to set a particular variable failed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>SNMP_REQUEST_GENERIC_ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Any error not covered by the other error types.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>SNMP_REQUEST_INCONSISTENT_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The variable does not exist; the agent cannot create it because the named object instance is inconsistent with the values of other managed objects.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>SNMP_REQUEST_INCONSISTENT_VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A variable binding specifies a value that could be held by the variable but cannot be assigned to it at this time.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>SNMP_REQUEST_NO_ACCESS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Access was denied to the object for security reasons.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>SNMP_REQUEST_NO_CREATION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A specified variable does not exist and cannot be created.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>SNMP_REQUEST_NO_SUCH_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The OID could not be found or there is no OID to respond to in a get next request.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>SNMP_REQUEST_NOT_WRITABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The variable does not exist; the agent cannot create it because the named object instance is inconsistent with the values of other managed objects.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>SNMP_REQUEST_READ_ONLY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An attempt was made to set a variable that has an access value of Read-Only</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>SNMP_REQUEST_ResourceUnavailable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An attempt to set a variable required a resource that is not available.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>SNMP_REQUEST_SUCCESS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The request completed successfully and the MIBObject contains valid contents.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>SNMP_REQUEST_TOO_BIG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The size of the Response-PDU would be too large to transport.</td>
</tr>
</tbody>
</table>
### Field Summary

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SNMP_REQUEST_UNDO_FAILED</strong></td>
<td>An attempt to set a particular variable as part of a group of variables failed, and the attempt to then undo the setting of other variables was not successful.</td>
</tr>
<tr>
<td><strong>SNMP_REQUEST_WRONG_ENCODING</strong></td>
<td>A variable binding specifies an encoding incorrect for the object.</td>
</tr>
<tr>
<td><strong>SNMP_REQUEST_WRONG_LENGTH</strong></td>
<td>A variable binding specifies a length incorrect for the object.</td>
</tr>
<tr>
<td><strong>SNMP_REQUEST_WRONG_TYPE</strong></td>
<td>The object type in the request is incorrect for the object.</td>
</tr>
<tr>
<td><strong>SNMP_REQUEST_WRONG_VALUE</strong></td>
<td>The value given in a variable binding is not possible for the object.</td>
</tr>
</tbody>
</table>

### Constructor Summary

**SNMPResponse(int status, MIBObject object)**

Constructs an SNMPResponse.

### Method Summary

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getMIBObject()</td>
<td>Gets the encoding of the MIB object associated with the OID in the request that caused this response.</td>
</tr>
<tr>
<td>getStatus()</td>
<td>Get the status of the response.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Field Detail

**SNMP_REQUEST_SUCCESS**

public static final int **SNMP_REQUEST_SUCCESS**

The request completed successfully and the MIBObject contains valid contents.

See Also:

Constant Field Values

**SNMP_REQUEST_TOO_BIG**

public static final int **SNMP_REQUEST_TOO_BIG**

The size of the Response-PDU would be too large to transport.

See Also:

Constant Field Values
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.
   See Also:
   Constant Field Values

SNMP_REQUEST_BAD_VALUE
public static final int SNMP_REQUEST_BAD_VALUE
   A Check/Set value (or syntax) error occurred.
   See Also:
   Constant Field Values

SNMP_REQUEST_READ_ONLY
public static final int SNMP_REQUEST_READ_ONLY
   An attempt was made to set a variable that has an access value of Read-Only
   See Also:
   Constant Field Values

SNMP_REQUEST_GENERIC_ERROR
public static final int SNMP_REQUEST_GENERIC_ERROR
   Any error not covered by the other error types.
   See Also:
   Constant Field Values

SNMP_REQUEST_NO_ACCESS
public static final int SNMP_REQUEST_NO_ACCESS
   Access was denied to the object for security reasons.
   See Also:
   Constant Field Values

SNMP_REQUEST_WRONG_TYPE
public static final int SNMP_REQUEST_WRONG_TYPE
   The object type in the request is incorrect for the object.
   See Also:
   Constant Field Values

SNMP_REQUEST_WRONG_LENGTH
public static final int SNMP_REQUEST_WRONG_LENGTH
   A variable binding specifies a length incorrect for the object.
   See Also:
   Constant Field Values

SNMP_REQUEST_WRONG_ENCODING
public static final int SNMP_REQUEST_WRONG_ENCODING
   A variable binding specifies an encoding incorrect for the object.
   See Also:
   Constant Field Values
SNMP_REQUEST_WRONG_VALUE
public static final int SNMP_REQUEST_WRONG_VALUE
The value given in a variable binding is not possible for the object
See Also:
Constant Field Values

SNMP_REQUEST_NO_CREATION
public static final int SNMP_REQUEST_NO_CREATION
A specified variable does not exist and cannot be created.
See Also:
Constant Field Values

SNMP_REQUEST_INCONSISTENT_VALUE
public static final int SNMP_REQUEST_INCONSISTENT_VALUE
A variable binding specifies a value that could be held by the variable but cannot be assigned to it at this
time. (For example, is not CURRENTLY valid to set because of the value of another MIB object, e.g., one
MIB value indicates if a clock display is 12 or 24 hours, and is set to 12, but then someone tries to set the
time to 13:00)
See Also:
Constant Field Values

SNMP_REQUEST_RESOURCE_UNAVAILABLE
public static final int SNMP_REQUEST_RESOURCE_UNAVAILABLE
An attempt to set a variable required a resource that is not available.
See Also:
Constant Field Values

SNMP_REQUEST_COMMIT_FAILED
public static final int SNMP_REQUEST_COMMIT_FAILED
An attempt to set a particular variable failed.
See Also:
Constant Field Values

SNMP_REQUEST_UNDO_FAILED
public static final int SNMP_REQUEST_UNDO_FAILED
An attempt to set a particular variable as part of a group of variables failed, and the attempt to then undo the
setting of other variables was not successful.
See Also:
Constant Field Values

SNMP_REQUEST_AUTHORIZATION_ERROR
public static final int SNMP_REQUEST_AUTHORIZATION_ERROR
A problem occurred in authorization.
See Also:
Constant Field Values
SNMP_REQUEST_NOT_WRITABLE
public static final int SNMP_REQUEST_NOT_WRITABLE
   The variable does not exist; the agent cannot create it because the named object instance is inconsistent with
   the values of other managed objects.
See Also:
   Constant Field Values

SNMP_REQUEST_INCONSISTENT_NAME
public static final int SNMP_REQUEST_INCONSISTENT_NAME
   The variable does not exist; the agent cannot create it because the named object instance is inconsistent with
   the values of other managed objects.
See Also:
   Constant Field Values

Constructor Detail

SNMPResponse
public SNMPResponse(int status,
                     MIBObject object)
   Constructs an SNMPResponse.
Parameters:
   status - Status of the corresponding request. Possible values include any of the SNMP_REQUEST_*
             constants in this class.
   object - MIBObject resulting from the corresponding request.

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 thegetStatus method returns SNMP_REQUEST_SUCCESS this method SHALL return a populated MIB
   Object, otherwise the MIB object returned SHALL contain an empty data array with length 0.
Annex AA Logger API

Only those Packages, Classes, and Interfaces created or modified by OCAP are shown in this annex.

Package org.ocap.logging

Class Summary

| MIBAppender | The MIBAppender appends events to a circular queue which is made available at a given OID. |

org.ocap.logging

Class MIBAppender

```java
public class MIBAppender
    extends AppenderSkeleton
```

The MIBAppender appends events to a circular queue which is made available at a given OID. The [MIB-HOST] specification defines a table specifically for this purpose, for which an application can use the OID of the ocStbHostSystemLoggingTable.

Constructor Summary

| MIBAppender() |

Method Summary

| void append(LoggingEvent event) Append a logging event to the MIB Appender. |
| void close() Closes the MIB appender. |
| Level getSize() |
| java.lang.String getOid() |
| int getSize() |
| boolean requiresLayout() Accessor reporting if MIBAppender requires a layout |
**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>setLevelMin(Level levelMin)</code></td>
<td>Set the minimum priority level that this Appender will actually execute for.</td>
</tr>
<tr>
<td>void</td>
<td><code>setOid(java.lang.String oid)</code></td>
<td>Sets the OID that this MIBAppender should be attached to.</td>
</tr>
<tr>
<td>void</td>
<td><code>setSize(int size)</code></td>
<td>Set the size of the circular queue in number of messages.</td>
</tr>
</tbody>
</table>

**Methods inherited from class org.apache.log4j.AppenderSkeleton**

- activateOptions
- addFilter
- clearFilters
- doAppend
- finalize
- getErrorHandler
- getFilter
- getFirstFilter
- getLayout
- getName
- getThreshold
- isAsSevereAsThreshold
- setErrorHandler
- setLayout
- setName
- setThreshold

**Methods inherited from class java.lang.Object**

- clone
- equals
- getClass
- hashCode
- notify
- notifyAll
- toString
- wait
- wait

**Constructor Detail**

**MIBAppender**

```java
public MIBAppender()
```

**Method Detail**

**close**

```java
public void close()
```

Closes the MIB appender.

**append**

```java
public void append(LoggingEvent event)
```

Appends a logging event to the MIB Appender. If the OID set for this MIB Appender is not an OID to a known circular logging table this method does nothing successfully.

**setOid**

```java
public void setOid(java.lang.String oid)
```

Sets the OID that this MIBAppender should be attached to. This is a required value for the `MIBAppender` and there is no default.

**Parameters:**

- `oid` - The OID for the MIB node where the queue of logger messages will be accessible.

**getOid**

```java
public java.lang.String getOid()
```
**setSize**

```java
public void setSize(int size)
```

Set the size of the circular queue in number of messages. (Default: 10)

**Parameters:**
- `size` - the size to make the circular queue

**getLevelMin**

```java
public Level getLevelMin()
```

**requiresLayout**

```java
public boolean requiresLayout()
```

Accessor reporting if MIBAppender requires a layout

**Returns:**
- `true`
Package org.apache.log4j

The main log4j package.

Class Summary

<table>
<thead>
<tr>
<th>Category</th>
<th>This class has been deprecated and replaced by the Logger subclass.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchy</td>
<td>This class is specialized in retrieving loggers by name and also maintaining the logger hierarchy.</td>
</tr>
<tr>
<td>Logger</td>
<td>This is the central class in the log4j package.</td>
</tr>
<tr>
<td>LogManager</td>
<td>Use the LogManager class to retrieve Logger instances or to operate on the current LoggerRepository.</td>
</tr>
<tr>
<td>PropertyConfigurator</td>
<td>Allows the configuration of log4j from an external file.</td>
</tr>
</tbody>
</table>

org.apache.log4j

Class Category

java.lang.Object

org.apache.log4j.Category

All Implemented Interfaces:

AppenderAttachable

Direct Known Subclasses:

Logger

public class Category
extends java.lang.Object
implements AppenderAttachable

This class has been deprecated and replaced by the Logger subclass. It will be kept around to preserve backward compatibility until mid 2003.

Logger is a subclass of Category, i.e. it extends Category. In other words, a logger is a category. Thus, all operations that can be performed on a category can be performed on a logger. Internally, whenever log4j is asked to produce a Category object, it will instead produce a Logger object. Log4j 1.2 will never produce Category objects but only Logger instances. In order to preserve backward compatibility, methods that previously accepted category objects still continue to accept category objects.

For example, the following are all legal and will work as expected.

```
// Deprecated form:
Category cat = Category.getInstance("foo.bar")

// Preferred form for retrieving loggers:
Logger logger = Logger.getLogger("foo.bar")
```

The first form is deprecated and should be avoided.

There is absolutely no need for new client code to use or refer to the Category class. Whenever possible, please avoid referring to it or using it.
See the short manual for an introduction on this class.

See the document entitled preparing for log4j 1.3 for a more detailed discussion.

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Category()</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
</table>
| **void addAppender(Appender newAppender)**  
Add newAppender to the list of appenders of this Category instance. |
| **void assertLog(boolean assertion, java.lang.String msg)**  
If assertion parameter is false, then logs msg as an error statement. |
| **void callAppenders(LoggingEvent event)**  
Call the appenders in the hierarchy starting at this. |
| **void debug(java.lang.Object message)**  
Log a message object with the DEBUG level. |
| **void debug(java.lang.Object message, java.lang.Throwable t)**  
Log a message object with the DEBUG level including the stack trace of the Throwable t passed as parameter. |
| **void error(java.lang.Object message)**  
Log a message object with the ERROR Level. |
| **void error(java.lang.Object message, java.lang.Throwable t)**  
Log a message object with the ERROR level including the stack trace of the Throwable t passed as parameter. |
| **static Logger exists(java.lang.String name)**  
Deprecated. Please use LogManager.exists(java.lang.String) instead. |
| **void fatal(java.lang.Object message)**  
Log a message object with the FATAL Level. |
| **void fatal(java.lang.Object message, java.lang.Throwable t)**  
Log a message object with the FATAL level including the stack trace of the Throwable t passed as parameter. |
| **boolean getAdditivity()**  
Get the additivity flag for this Category instance. |
| **java.utilEnumeration getAllAppenders()**  
Get the appenders contained in this category as an Enumeration. |
Method Summary

Appender

- `getAppender(java.lang.String name)`
  Look for the appender named as name.

Priority

- `getChainedPriority()`
  Deprecated. Please use the `getEffectiveLevel()` method instead.

static `java.util Enumeration$currentCategories()`

- `getCurrentCategories()`
  Deprecated. Please use `LogManager.getCurrentLoggers()` instead.

static `LoggerRepository$getDefaultHierarchy()`

- `getDefaultHierarchy()`
  Deprecated. Please use `LogManager.getLoggerRepository()` instead.

Level

- `getEffectiveLevel()`
  Starting from this category, search the category hierarchy for a non-null level and return it.

LoggerRepository

- `getHierarchy()`
  Deprecated. Please use `getLoggerRepository()` instead.

static Category

- `getInstance(java.lang.Class clazz)`
  Deprecated. Please make sure to use `Logger.getLogger(Class)` instead.

static Category

- `getInstance(java.lang.String name)`
  Deprecated. Make sure to use `Logger.getLogger(String)` instead.

Level

- `getLevel()`
  Returns the assigned Level, if any, for this Category.

LoggerRepository

- `getLoggerRepository()`
  Return the `LoggerRepository` where this Category is attached.

java.lang.String

- `getName()`
  Return the category name.

Category

- `getParent()`
  Returns the parent of this category.

Level

- `getPriority()`
  Deprecated. Please use `getLevel()` instead.

java.util.ResourceBundle

- `getResourceBundle()`
  Return the inherited `ResourceBundle` for this category.

static Category

- `getRoot()`
  Deprecated. Please use `Logger.getRootLogger()` instead.

void

- `info(java.lang.Object message)`
  Log a message object with the INFO Level.

void

- `info(java.lang.Object message, java.lang.Throwable t)`
  Log a message object with the INFO level including the stack trace of the `Throwable` t passed as parameter.
## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>boolean isAttached(Appender appender)</code></td>
<td>Is the appender passed as parameter attached to this category?</td>
</tr>
<tr>
<td><code>boolean isDebugEnabled()</code></td>
<td>Check whether this category is enabled for the DEBUG Level.</td>
</tr>
<tr>
<td><code>boolean isEnabledFor(Priority level)</code></td>
<td>Check whether this category is enabled for a given Level passed as parameter.</td>
</tr>
<tr>
<td><code>boolean isErrorEnabled()</code></td>
<td>Check whether this category is enabled for the ERROR Level.</td>
</tr>
<tr>
<td><code>boolean isEnabledFor(Priority level)</code></td>
<td>Check whether this category is enabled for a given Level passed as parameter.</td>
</tr>
<tr>
<td><code>boolean isFatalEnabled()</code></td>
<td>Check whether this category is enabled for the FATAL Level.</td>
</tr>
<tr>
<td><code>boolean isInfoEnabled()</code></td>
<td>Check whether this category is enabled for the info Level.</td>
</tr>
<tr>
<td><code>boolean isWarnEnabled()</code></td>
<td>Check whether this category is enabled for the WARN Level.</td>
</tr>
<tr>
<td><code>void log(Priority priority, java.lang.String callerFQCN, Priority level, java.lang.Object message, java.lang.Throwable t)</code></td>
<td>This is the most generic printing method.</td>
</tr>
<tr>
<td><code>void log(Priority priority, java.lang.String key, java.lang.Throwable t)</code></td>
<td>Log a localized message.</td>
</tr>
<tr>
<td><code>void log(Priority priority, java.lang.Object message)</code></td>
<td>This generic form is intended to be used by wrappers.</td>
</tr>
<tr>
<td><code>void removeAppender(String name)</code></td>
<td>Remove the appender with the name passed as parameter from the list of appenders.</td>
</tr>
<tr>
<td><code>void removeAppender(Appender appender)</code></td>
<td>Remove the appender passed as parameter form the list of appenders.</td>
</tr>
<tr>
<td><code>void removeAllAppenders()</code></td>
<td>Remove all previously added appenders from this Category instance.</td>
</tr>
<tr>
<td><code>void setAdditivity(boolean additive)</code></td>
<td>Set the additivity flag for this Category instance.</td>
</tr>
<tr>
<td><code>void setLevel(Level level)</code></td>
<td>Set the level of this Category.</td>
</tr>
</tbody>
</table>
Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void setPriority(Priority priority)</code></td>
<td><code>setLevel(org.apache.log4j.Level) instead.</code></td>
</tr>
</tbody>
</table>
| `void setResourceBundle(java.util.ResourceBundle bundle)` | Set the resource bundle to be used with localized logging methods `l7dlog(Priority,String,Throwable)` and `l7dlog(Priority,String,Object[],Throwable)`.
| `static void shutdown()` | ` LogManager.shutdown() instead.`                                      |
| `void warn(java.lang.Object message)` | Log a message object with the WARN Level.                                  |
| `void warn(java.lang.Object message, java.lang.Throwable t)` | Log a message with the WARN level including the stack trace of the Throwable t passed as parameter. |

Methods inherited from class java.lang.Object

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`  

Constructor Detail

**Category**

```java
public Category()
```

Method Detail

**addAppender**

```java
public void addAppender(Appender newAppender)
```

Add `newAppender` to the list of appenders of this Category instance. If `newAppender` is already in the list of appenders, then it won't be added again.

*Specified by:* `addAppender in interface AppenderAttachable`

**assertLog**

```java
public void assertLog(boolean assertion, java.lang.String msg)
```

If `assertion` parameter is false, then logs `msg` as an error statement.

The `assert` method has been renamed to `assertLog` because `assert` is a language reserved word in JDK 1.4.

*Parameters:*

- `assertion`
- `msg` - The message to print if `assertion` is false.
Since:
1.2

callAppenders
public void callAppenders(LoggingEvent event)
Call the appenders in the hierarchy starting at this. If no appenders could be found, emit a warning.
This method calls all the appenders inherited from the hierarchy circumventing any evaluation of whether to log or not to log the particular log request.

Parameters:
event - the event to log.

debug
public void debug(java.lang.Object message)
Log a message object with the DEBUG level.
This method first checks if this category is DEBUG enabled by comparing the level of this category with the DEBUG level. If this category is DEBUG enabled, then it converts the message object (passed as parameter) to a string by invoking the appropriate ObjectRenderer. It then proceeds to call all the registered appenders in this category and also higher in the hierarchy depending on the value of the additivity flag.

WARNING Note that passing a Throwable to this method will print the name of the Throwable but no stack trace. To print a stack trace use the debug(Object, Throwable) form instead.

Parameters:
message - the message object to log.

debug
public void debug(java.lang.Object message, java.lang.Throwable t)
Log a message object with the DEBUG level including the stack trace of the Throwable t passed as parameter.
See debug(Object) form for more detailed information.

Parameters:
message - the message object to log.
t - the exception to log, including its stack trace.

error
public void error(java.lang.Object message)
Log a message object with the ERROR Level.
This method first checks if this category is ERROR enabled by comparing the level of this category with ERROR Level. If this category is ERROR enabled, then it converts the message object passed as parameter to a string by invoking the appropriate ObjectRenderer. It proceeds to call all the registered appenders in this category and also higher in the hierarchy depending on the value of the additivity flag.

WARNING Note that passing a Throwable to this method will print the name of the Throwable but no stack trace. To print a stack trace use the error(Object, Throwable) form instead.

Parameters:
message - the message object to log
error

class com.sun.jdi.debugInterface.Error

public void error(Object message, Throwable t)

Log a message object with the ERROR level including the stack trace of the Throwable t passed as parameter.

See error(Object) form for more detailed information.

Parameters:
message - the message object to log.
t - the exception to log, including its stack trace.

exists

class com.sun.jdi.debugInterface.Error.

public static Logger exists(String name)

Deprecated. Please use LogManager.exists(String) instead.

If the named category exists (in the default hierarchy) then it returns a reference to the category, otherwise it returns null.

Since:
0.8.5

fatal

class com.sun.jdi.debugInterface.Error.

public void fatal(Object message)

Log a message object with the FATAL Level.

This method first checks if this category is FATAL enabled by comparing the level of this category with FATAL Level. If the category is FATAL enabled, then it converts the message object passed as parameter to a string by invoking the appropriate ObjectRenderer. It proceeds to call all the registered appenders in this category and also higher in the hierarchy depending on the value of the additivity flag.

WARNING Note that passing a Throwable to this method will print the name of the Throwable but no stack trace. To print a stack trace use the fatal(Object, Throwable) form instead.

Parameters:
message - the message object to log.

public void fatal(Object message, Throwable t)

Log a message object with the FATAL level including the stack trace of the Throwable t passed as parameter.

See fatal(Object) for more detailed information.

Parameters:
message - the message object to log.
t - the exception to log, including its stack trace.

getAdditivity

class com.sun.jdi.debugInterface.Error.

public boolean getAdditivity()

Get the additivity flag for this Category instance.

getAllAppenders

class com.sun.jdi.debugInterface.Error.

public java.util.Enumeration getAllAppenders()
Get the appenders contained in this category as an Enumeration. If no appenders can be found, then a NullEnumeration is returned.

Specified by:
getAppenders in interface AppenderAttachable
Returns:
Enumeration An enumeration of the appenders in this category.

getAppender

public Appender getAppender(java.lang.String name)

Look for the appender named as name.

Return the appender with that name if in the list. Return null otherwise.

Specified by:
getAppender in interface AppenderAttachable

ggetEffectiveLevel

public Level getEffectiveLevel()

Starting from this category, search the category hierarchy for a non-null level and return it. Otherwise, return the level of the root category.

The Category class is designed so that this method executes as quickly as possible.

ggetChainedPriority

public Priority getChainedPriority()

Deprecated. Please use the getEffectiveLevel() method instead.

ggetCurrentCategories

public static java.util.Enumeration getCurrentCategories()

Deprecated. Please use LogManager.getCurrentLoggers() instead.

Returns all the currently defined categories in the default hierarchy as an Enumeration.

The root category is not included in the returned Enumeration.

ggetDefaultHierarchy

public static LoggerRepository getDefaultHierarchy()

Deprecated. Please use LogManager.getLoggerRepository() instead.

Return the default Hierarchy instance.

Since:
1.0

ggetHierarchy

public LoggerRepository getHierarchy()

Deprecated. Please use getLoggerRepository() instead.

Return the Hierarchy where this Category instance is attached.

Since:
1.1
**getLoggerRepository**

public LoggerRepository getLoggerRepository()

    Return the LoggerRepository where this Category is attached.

    Since:
    1.2

**getInstance**

public static Category getInstance(java.lang.String name)

    Deprecated. Make sure to use Logger.getLogger(String) instead.

getInstance

public static Category getInstance(java.lang.Class clazz)

    Deprecated. Please make sure to use Logger.getLogger(Class) instead.

**getName**

public final java.lang.String getName()

    Return the category name.

**getParent**

public final Category getParent()

    Returns the parent of this category. Note that the parent of a given category may change during the lifetime of the category.

    The root category will return null.

    Since:
    1.2

**getLevel**

public final Level getLevel()

    Returns the assigned Level, if any, for this Category.

    Returns:
    Level - the assigned Level, can be null.

**getPriority**

public final Level getPriority()

    Deprecated. Please use getLevel() instead.

**getRoot**

public static final Category getRoot()

    Deprecated. Please use Logger.getRootLogger() instead.

**getResourceBundle**

public java.util.ResourceBundle getResourceBundle()

    Return the inherited ResourceBundle for this category.
This method walks the hierarchy to find the appropriate resource bundle. It will return the resource bundle attached to the closest ancestor of this category, much like the way priorities are searched. In case there is no bundle in the hierarchy then null is returned.

Since:
0.9.0

info

public void info(java.lang.Object message)
Log a message object with the INFO Level.

This method first checks if this category is INFO enabled by comparing the level of this category with INFO Level. If the category is INFO enabled, then it converts the message object passed as parameter to a string by invoking the appropriate ObjectRenderer. It proceeds to call all the registered appenders in this category and also higher in the hierarchy depending on the value of the additivity flag.

WARNING Note that passing a Throwable to this method will print the name of the Throwable but no stack trace. To print a stack trace use the info(Object, Throwable) form instead.

Parameters:
message - the message object to log

info

public void info(java.lang.Object message, java.lang.Throwable t)
Log a message object with the INFO level including the stack trace of the Throwable t passed as parameter.

See info(Object) for more detailed information.

Parameters:
message - the message object to log.
t - the exception to log, including its stack trace.

isAttached

public boolean isAttached(Appender appender)
Is the appender passed as parameter attached to this category?

Specified by:
isAttached in interface AppenderAttachable

isDebugEnabled

public boolean isDebugEnabled()
Check whether this category is enabled for the DEBUG Level.

This function is intended to lessen the computational cost of disabled log debug statements.

For some cat Category object, when you write,

cat.debug("This is entry number: " + i);

You incur the cost constructing the message, concatenation in this case, regardless of whether the message is logged or not.

If you are worried about speed, then you should write
if (cat.isDebugEnabled()) {
    cat.debug("This is entry number: " + i);
}

This way you will not incur the cost of parameter construction if debugging is disabled for cat. On the other hand, if the cat is debug enabled, you will incur the cost of evaluating whether the category is debug enabled twice. Once in isDebugEnabled and once in the debug. This is an insignificant overhead since evaluating a category takes about 1% of the time it takes to actually log.

Returns:
boolean - true if this category is debug enabled, false otherwise.

isWarnEnabled

public boolean isWarnEnabled()
Check whether this category is enabled for the WARN Level.
This function is intended to lessen the computational cost of disabled log warn statements.

For some cat Category object, when you write,

cat.warn("This is entry number: " + i);

You incur the cost constructing the message, concatenation in this case, regardless of whether the message is logged or not.

If you are worried about speed, then you should write

    if (cat.isWarnEnabled()) {
        cat.warn("This is entry number: " + i);
    }

This way you will not incur the cost of parameter construction if warn is disabled for cat. On the other hand, if the cat is warn enabled, you will incur the cost of evaluating whether the category is warn enabled twice. Once in isWarnEnabled and once in the warn. This is an insignificant overhead since evaluating a category takes about 1% of the time it takes to actually log.

Returns:
boolean - true if this category is warn enabled, false otherwise.

isErrorEnabled

public boolean isErrorEnabled()
Check whether this category is enabled for the ERROR Level.
This function is intended to lessen the computational cost of disabled log error statements.

For some cat Category object, when you write,

cat.error("This is entry number: " + i);

You incur the cost constructing the message, concatenation in this case, regardless of whether the message is logged or not.

If you are worried about speed, then you should write
if(cat.isErrorEnabled()) {
    cat.error("This is entry number: " + i);
}

This way you will not incur the cost of parameter construction if error is disabled for cat. On the other hand, if the cat is warn enabled, you will incur the cost of evaluating whether the category is error enabled twice. Once in isErrorEnabled and once in the error. This is an insignificant overhead since evaluating a category takes about 1%% of the time it takes to actually log.

**Returns:**
boolean - true if this category is error enabled, false otherwise.

### isFatalEnabled

```java
public boolean isFatalEnabled()
```

Check whether this category is enabled for the FATAL Level.

This function is intended to lessen the computational cost of disabled log fatal statements.

For some `cat` Category object, when you write,

```java
cat.fatal("This is entry number: " + i);
```

You incur the cost constructing the message, concatenation in this case, regardless of whether the message is logged or not.

If you are worried about speed, then you should write

```java
if(cat.isDebugEnabled()) {
    cat.fatal("This is entry number: " + i);
}
```

This way you will not incur the cost of parameter construction if fatal is disabled for cat. On the other hand, if the cat is fatal enabled, you will incur the cost of evaluating whether the category is fatal enabled twice. Once in isFatalEnabled and once in the fatal. This is an insignificant overhead since evaluating a category takes about 1%% of the time it takes to actually log.

**Returns:**
boolean - true if this category is fatal enabled, false otherwise.

### isEnabledFor

```java
public boolean isEnabledFor(Priority level)
```

Check whether this category is enabled for a given Level passed as parameter. See also isDebugEnabled().

**Returns:**
boolean True if this category is enabled for level.

### isInfoEnabled

```java
public boolean isInfoEnabled()
```

Check whether this category is enabled for the info Level. See also isDebugEnabled().

**Returns:**
boolean - true if this category is enabled for level info, false otherwise.
**l7dlog**

```java
public void l7dlog(Priority priority,
                   java.lang.String key,
                   java.lang.Throwable t)
```

Log a localized message. The user supplied parameter `key` is replaced by its localized version from the resource bundle.

**Since:**
0.8.4

**See Also:**
setResourceBundle(java.util.ResourceBundle)

---

**l7dlog**

```java
public void l7dlog(Priority priority,
                   java.lang.String key,
                   java.lang.Object[] params,
                   java.lang.Throwable t)
```

Log a localized and parameterized message. First, the user supplied `key` is searched in the resource bundle. Next, the resulting pattern is formatted using `java.text.MessageFormat#format(String,Object[])` method with the user supplied object array `params`.

**Since:**
0.8.4

---

**log**

```java
public void log(Priority priority,
                java.lang.Object message,
                java.lang.Throwable t)
```

This generic form is intended to be used by wrappers.

---

**log**

```java
public void log(Priority priority,
                java.lang.Object message)
```

This generic form is intended to be used by wrappers.

---

**log**

```java
public void log(java.lang.String callerFQCN,
                Priority level,
                java.lang.Object message,
                java.lang.Throwable t)
```

This is the most generic printing method. It is intended to be invoked by wrapper classes.

**Parameters:**
callerFQCN - The wrapper class' fully qualified class name.
level - The level of the logging request.
message - The message of the logging request.
t - The throwable of the logging request, may be null.

---

**removeAllAppenders**

```java
public void removeAllAppenders()
```

Remove all previously added appenders from this Category instance.
This is useful when re-reading configuration information.

Specified by:
removeAllAppenders in interface AppenderAttachable

removeAppender

public void removeAppender(Appender appender)
Remove the appender passed as parameter form the list of appenders.
Specified by:
removeAppender in interface AppenderAttachable
Since:
0.8.2

removeAppender

public void removeAppender(java.lang.String name)
Remove the appender with the name passed as parameter form the list of appenders.
Specified by:
removeAppender in interface AppenderAttachable
Since:
0.8.2

setAdditivity

public void setAdditivity(boolean additive)
Set the additivity flag for this Category instance.
Since:
0.8.1

setLevel

public void setLevel(Level level)
Set the level of this Category. If you are passing any of Level.DEBUG, Level.INFO, Level.WARN, Level.ERROR, Level.FATAL as a parameter, you need to case them as Level.
As in
logger.setLevel((Level) Level.DEBUG);
Null values are admitted.

setPriority

public void setPriority(Priority priority)
Deprecated. Please use setLevel(org.apache.log4j.Level) instead.
Set the level of this Category.
Null values are admitted.

setResourceBundle

public void setResourceBundle(java.util.ResourceBundle bundle)
Set the resource bundle to be used with localized logging methods
l7dlog(Priority, String, Throwable) and
l7dlog(Priority, String, Object[], Throwable).
Since:
0.8.4

shutdown

public static void shutdown()

*Deprecated.* Please use LogManager.shutdown() instead.

Calling this method will safely close and remove all appenders in all the categories including root contained in the default hierarchy.

Some appenders such as org.apache.log4j.net.SocketAppender and AsyncAppender need to be closed before the application exists. Otherwise, pending logging events might be lost.

The shutdown method is careful to close nested appenders before closing regular appenders. This is allows configurations where a regular appender is attached to a category and again to a nested appender.

Since:
1.0

warn

public void warn(java.lang.Object message)

Log a message object with the WARN Level.

This method first checks if this category is WARN enabled by comparing the level of this category with WARN Level. If the category is WARN enabled, then it converts the message object passed as parameter to a string by invoking the appropriate ObjectRenderer. It proceeds to call all the registered appenders in this category and also higher in the hierarchy depending on the value of the additivity flag.

WARNING Note that passing a Throwable to this method will print the name of the Throwable but no stack trace. To print a stack trace use the warn(Object, Throwable) form instead.

Parameters:
message - the message object to log.

warn

public void warn(java.lang.Object message, java.lang.Throwable t)

Log a message with the WARN level including the stack trace of the Throwable t passed as parameter.

See warn(Object) for more detailed information.

Parameters:
message - the message object to log.
t - the exception to log, including its stack trace.
org.apache.log4j

Class Hierarchy

java.lang.Object
  └─org.apache.log4j.Hierarchy

All Implemented Interfaces:
  LoggerRepository, RendererSupport, ThrowableRendererSupport

public class Hierarchy
  extends java.lang.Object
  implements LoggerRepository, RendererSupport, ThrowableRendererSupport

This class is specialized in retrieving loggers by name and also maintaining the logger hierarchy.

*The casual user does not have to deal with this class directly.*

The structure of the logger hierarchy is maintained by the getLogger(java.lang.String, java.lang.String) method. The hierarchy is such that children link to their parent but parents do not have any pointers to their children. Moreover, loggers can be instantiated in any order, in particular descendant before ancestor.

In case a descendant is created before a particular ancestor, then it creates a provision node for the ancestor and adds itself to the provision node. Other descendants of the same ancestor add themselves to the previously created provision node.

### Constructor Summary

**Hierarchy(Logger root)**

Create a new logger hierarchy.

### Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th>addHierarchyEventListener(HierarchyEventListener listener)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Add a HierarchyEventListener event to the repository.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>addRenderer(java.lang.Class classToRender, ObjectRenderer or)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Add an object renderer for a specific class.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>clear()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This call will clear all logger definitions from the internal hashtable.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>emitNoAppenderWarning(Category cat)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Logger</th>
<th>exists(java.lang.String name)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Check if the named logger exists in the hierarchy.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>fireAddAppenderEvent(Category logger, Appender appender)</th>
</tr>
</thead>
</table>
## Method Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>java.util.Enumeration</code></td>
<td><code>getCurrentCategories()</code></td>
<td>Returns all the currently defined categories in this hierarchy as an Enumeration. <strong>Deprecated. Please use <code>getCurrentLoggers()</code> instead.</strong></td>
</tr>
<tr>
<td><code>java.util.Enumeration</code></td>
<td><code>getCurrentLoggers()</code></td>
<td>Returns all the currently defined categories in this hierarchy as an Enumeration.</td>
</tr>
<tr>
<td><code>Logger</code></td>
<td><code>getLogger(java.lang.String name)</code></td>
<td>Return a new logger instance named as the first parameter using the default factory.</td>
</tr>
<tr>
<td><code>Logger</code></td>
<td><code>getLogger(java.lang.String name, LoggerFactory factory)</code></td>
<td>Return a new logger instance named as the first parameter using factory.</td>
</tr>
<tr>
<td><code>Logger</code></td>
<td><code>getLogger(java.lang.String name, java.lang.String group)</code></td>
<td>Return a new logger instance named as the first parameter using the default factory.</td>
</tr>
<tr>
<td><code>Logger</code></td>
<td><code>getLogger(java.lang.String name, LoggerFactory factory, java.lang.String group)</code></td>
<td>Return a new logger instance named as the first parameter using factory.</td>
</tr>
<tr>
<td><code>RendererMap</code></td>
<td><code>getRendererMap()</code></td>
<td>Get the renderer map for this hierarchy.</td>
</tr>
<tr>
<td><code>Logger</code></td>
<td><code>getRootLogger()</code></td>
<td>Get the root of this hierarchy.</td>
</tr>
<tr>
<td><code>Level</code></td>
<td><code>getThreshold()</code></td>
<td>Returns a <code>Level</code> representation of the enable state.</td>
</tr>
<tr>
<td><code>ThrowableRenderer</code></td>
<td><code>getThrowableRenderer()</code></td>
<td>Get throwable renderer.</td>
</tr>
<tr>
<td><code>boolean</code></td>
<td><code>isDisabled(int level)</code></td>
<td>This method will return <code>true</code> if this repository is disabled for <code>level</code> object passed as parameter and <code>false</code> otherwise.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>overrideAsNeeded(java.lang.String override)</code></td>
<td></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>resetConfiguration()</code></td>
<td>Reset all values contained in this hierarchy instance to their default.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>setDisableOverride(java.lang.String override)</code></td>
<td><strong>Deprecated. Deprecated with no replacement.</strong></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>setGroup(java.lang.String groupName, Level groupThreshold)</code></td>
<td></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>setRenderer(java.lang.Class renderedClass, ObjectRenderer renderer)</code></td>
<td>Used by subclasses to add a renderer to the hierarchy passed as parameter.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>setThreshold(Level l)</code></td>
<td>Enable logging for logging requests with level <code>l</code> or higher.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>setThreshold(java.lang.String levelStr)</code></td>
<td>The string form of <code>setThreshold(Level).</code></td>
</tr>
</tbody>
</table>
Method Summary

| void setThrowableRenderer(ThrowableRenderer renderer) | Set throwable renderer. |
| void shutdown() | Shutting down a hierarchy will safely close and remove all appenders in all categories including the root logger. |

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Hierarchy

public Hierarchy(Logger root)
Create a new logger hierarchy.
Parameters:
root - The root of the new hierarchy.

Method Detail

addRenderer

public void addRenderer(java.lang.Class classToRender, ObjectRenderer or)
Add an object renderer for a specific class.

addHierarchyEventListener

public void addHierarchyEventListener(HierarchyEventListener listener)
Description copied from interface: LoggerRepository
Add a HierarchyEventListener event to the repository.
Specified by:
addHierarchyEventListener in interface LoggerRepository

clear

public void clear()
This call will clear all logger definitions from the internal hashtable. Invoking this method will irrevocably mess up the logger hierarchy.
You should really know what you are doing before invoking this method.
Since:
0.9.0

emitNoAppenderWarning

public void emitNoAppenderWarning(Category cat)
Specified by:
emitNoAppenderWarning in interface LoggerRepository
exists
public Logger exists(java.lang.String name)
    Check if the named logger exists in the hierarchy. If so return its reference, otherwise returns null.
    Specified by:
    exists in interface LoggerRepository
    Parameters:
    name - The name of the logger to search for.

setThreshold
public void setThreshold(java.lang.String levelStr)
    The string form of setThreshold-Level).
    Specified by:
    setThreshold in interface LoggerRepository

setThreshold
public void setThreshold(Level l)
    Enable logging for logging requests with level l or higher. By default all levels are enabled.
    Specified by:
    setThreshold in interface LoggerRepository
    Parameters:
    l - The minimum level for which logging requests are sent to their appenders.

fireAddAppenderEvent
public void fireAddAppenderEvent(Category logger,
        Appender appender)
    Specified by:
    fireAddAppenderEvent in interface LoggerRepository

getThreshold
public Level getThreshold()
    Returns a Level representation of the enable state.
    Specified by:
    getThreshold in interface LoggerRepository
    Since:
    1.2

getLogger
public Logger getLogger(java.lang.String name,
        java.lang.String group)
    Return a new logger instance named as the first parameter using the default factory.
    If a logger of that name already exists, then it will be returned. Otherwise, a new logger will be instantiated
    and then linked with its existing ancestors as well as children.
    Parameters:
    name - The name of the logger to retrieve.
setGroup

public void setGroup(java.lang.String groupName,
        Level groupThreshold)

getLogger

public Logger getLogger(java.lang.String name,
        LoggerFactory factory)

Specified by:
        getLogger in interface LoggerRepository

getLogger

public Logger getLogger(java.lang.String name,
        java.lang.String group,
        LoggerFactory factory)

Return a new logger instance named as the first parameter using factory.

If a logger of that name already exists, then it will be returned. Otherwise, a new logger will be instantiated by the factory parameter and linked with its existing ancestors as well as children.

Parameters:
        name - The name of the logger to retrieve.
        factory - The factory that will make the new logger instance.

getCurrentLoggers

public java.util.Enumeration getCurrentLoggers()

Returns all the currently defined categories in this hierarchy as an Enumeration.

The root logger is not included in the returned Enumeration.

Specified by:
        getCurrentLoggers in interface LoggerRepository

gGetCurrentCategories

public java.util.Enumeration getCurrentCategories()

Deprecated. Please use getCurrentLoggers() instead.

Description copied from interface LoggerRepository

Deprecated. Please use LoggerRepository.getCurrentLoggers() instead.

Specified by:
        getCurrentCategories in interface LoggerRepository

gerendererMap

public RendererMap getRendererMap()

Get the renderer map for this hierarchy.

Specified by:
        getRendererMap in interface RendererSupport

getRootLogger

public Logger getRootLogger()

Get the root of this hierarchy.

Specified by:
        getRootLogger in interface LoggerRepository
Since: 0.9.0

isDisabled

public boolean isDisabled(int level)
This method will return true if this repository is disabled for level object passed as parameter and false otherwise. See also the threshold method.
Specified by:
isDisabled in interface LoggerRepository

overrideAsNeeded

public void overrideAsNeeded(java.lang.String override)
Deprecated. Deprecated with no replacement.

resetConfiguration

public void resetConfiguration()
Reset all values contained in this hierarchy instance to their default. This removes all appenders from all categories, sets the level of all non-root categories to null, sets their additivity flag to true and sets the level of the root logger to DEBUG. Moreover, message disabling is set its default "off" value.

Existing categories are not removed. They are just reset.

This method should be used sparingly and with care as it will block all logging until it is completed.
Specified by:
resetConfiguration in interface LoggerRepository
Since: 0.8.5

getLogger

public Logger getLogger(java.lang.String name)
Specified by:
getLogger in interface LoggerRepository

setDisableOverride

public void setDisableOverride(java.lang.String override)
Deprecated. Deprecated with no replacement.
Does nothing.

setRenderer

public void setRenderer(java.lang.Class renderedClass,
ObjectRenderer renderer)
Used by subclasses to add a renderer to the hierarchy passed as parameter.
Specified by:
setRenderer in interface RendererSupport

setThrowableRenderer

public void setThrowableRenderer(ThrowableRenderer renderer)
Set throwable renderer.
Specified by:
setThrowableRenderer in interface ThrowableRendererSupport
Parameters:
renderer - renderer, may be null.

getThrowableRenderer
public ThrowableRenderer getThrowableRenderer()
Get throwable renderer.
Specified by:
getThrowableRenderer in interface ThrowableRendererSupport
Returns:
throwable renderer, may be null.

shutdown
public void shutdown()
Shutting down a hierarchy will safely close and remove all appenders in all categories including the root logger.

Some appenders such as org.apache.log4j.net.SocketAppender and AsyncAppender need to be closed before the application exists. Otherwise, pending logging events might be lost.

The shutdown method is careful to close nested appenders before closing regular appenders. This is allows configurations where a regular appender is attached to a logger and again to a nested appender.

Specified by:
shutdown in interface LoggerRepository
Since:
1.0
org.apache.log4j
Class Logger

java.lang.Object
   org.apache.log4j.Category
   org.apache.log4j.Logger

All Implemented Interfaces:
   AppenderAttachable

public class Logger
extends Category

This is the central class in the log4j package. Most logging operations, except configuration, are done through this class.

NOTE: This class is based on the standard log4j 1.2 Logger class with additional support for associating Loggers in order to simplify configuration of groups of Loggers, as well as the string concatenation-avoidance helper methods from the LogSF utility class.

Support for Logger 'groups'

The common practice of using a fully qualified class name as the argument to getLogger(String) simplifies configuration, as inherited Loggers share a Level threshold and appender configuration unless explicitly overridden. However, Loggers may be functionally related but not part of a common package structure, requiring multiple entries in the log4j configuration file to configure these related Loggers with the same Level threshold.

To support the ability to easily configure the verbosity threshold of functionally-related Loggers, the getLogger(java.lang.String) methods have been overloaded to include versions which accept a group name parameter. Groups act as an 'alias' for controlling the verbosity threshold of a number of related Loggers.

If a Logger is retrieved using one of the forms which accepts a group name, the group name and an associated level can be defined in the log4j configuration file. All loggers associated with that group name will log at the specified level or greater severity.

The root logger logging severity threshold can be overridden by a group definition, which can be overridden by an explicit logger definition.

Groups support is implemented by setting the Logger's 'effective level' to the group level if applicable, causing isEnabled methods to return true based on the group severity threshold if applicable.

See PropertyConfigurator and DOMConfigurator for information on how to specify a group definition for properties-formatted and XML-formatted log4j configuration files.

Parameterized messages

The string concatenation-avoidance helper methods from the LogSF utility class have been incorporated into the Logger class, supporting parameterized messages and simplifying usage of the Logging API.

These methods overload the level-specific logging methods with additional forms supporting a number of commonly-needed arguments, delaying string concatenation until after the level severity threshold check. All level-specific logging methods also provide a form accepting an array of Objects, allowing any number of replacements.

Format of parameterized messages
The first argument of the forms supporting parameterized messages is a String which can contain pairs of braces {} as placeholders where the additional method arguments will be placed, in order, after the level severity threshold check has passed.

Example: log.info("Found {} matches", matchCount);

Since: log4j 1.2

### Constructor Summary

**Logger()**

### Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th>debug(java.lang.String pattern, boolean argument)</th>
</tr>
</thead>
</table>
| Log a parameterized message at debug level.

<table>
<thead>
<tr>
<th>void</th>
<th>debug(java.lang.String pattern, byte argument)</th>
</tr>
</thead>
</table>
| Log a parameterized message at debug level.

<table>
<thead>
<tr>
<th>void</th>
<th>debug(java.lang.String pattern, char argument)</th>
</tr>
</thead>
</table>
| Log a parameterized message at debug level.

<table>
<thead>
<tr>
<th>void</th>
<th>debug(java.lang.String pattern, double argument)</th>
</tr>
</thead>
</table>
| Log a parameterized message at debug level.

<table>
<thead>
<tr>
<th>void</th>
<th>debug(java.lang.String pattern, float argument)</th>
</tr>
</thead>
</table>
| Log a parameterized message at debug level.

<table>
<thead>
<tr>
<th>void</th>
<th>debug(java.lang.String pattern, int argument)</th>
</tr>
</thead>
</table>
| Log a parameterized message at debug level.

<table>
<thead>
<tr>
<th>void</th>
<th>debug(java.lang.String pattern, long argument)</th>
</tr>
</thead>
</table>
| Log a parameterized message at debug level.

<table>
<thead>
<tr>
<th>void</th>
<th>debug(java.lang.String pattern, java.lang.Object[] arguments)</th>
</tr>
</thead>
</table>
| Log a parameterized message at debug level.

<table>
<thead>
<tr>
<th>void</th>
<th>debug(java.lang.String pattern, java.lang.Object arg0, java.lang.Object arg1)</th>
</tr>
</thead>
</table>
| Log a parameterized message at debug level.

<table>
<thead>
<tr>
<th>void</th>
<th>debug(java.lang.String pattern, java.lang.Object arg0, java.lang.Object arg1, java.lang.Object arg2)</th>
</tr>
</thead>
</table>
| Log a parameterized message at debug level.

|------|--------------------------------------------------------------------------------------------------------------------------|
| Log a parameterized message at debug level.

<table>
<thead>
<tr>
<th>void</th>
<th>debug(java.lang.String pattern, java.lang.Object argument, java.lang.Throwable t)</th>
</tr>
</thead>
</table>
| Log a parameterized message at debug level.

<table>
<thead>
<tr>
<th>void</th>
<th>debug(java.lang.String pattern, short argument)</th>
</tr>
</thead>
</table>
| Log a parameterized message at debug level.
## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug</code></td>
<td><code>void debug(java.lang.Throwable t, java.lang.String pattern, java.lang.Object[] arguments)</code></td>
<td>Log a parameterized message at debug level.</td>
</tr>
<tr>
<td><code>error</code></td>
<td><code>void error(java.lang.String pattern, java.lang.Object[] arguments)</code></td>
<td>Log a parameterized message at error level.</td>
</tr>
<tr>
<td><code>fatal</code></td>
<td><code>void fatal(java.lang.String pattern, java.lang.Object[] arguments)</code></td>
<td>Log a parameterized message at fatal level.</td>
</tr>
<tr>
<td><code>fatal</code></td>
<td><code>void fatal(java.lang.Throwable t, java.lang.String pattern, java.lang.Object[] arguments)</code></td>
<td>Log a parameterized message at fatal level.</td>
</tr>
<tr>
<td><code>getLogger</code></td>
<td><code>static Logger getLogger(java.lang.Class clazz)</code></td>
<td>Shorthand for <code>getLogger(clazz.getName())</code>.</td>
</tr>
<tr>
<td><code>getLogger</code></td>
<td><code>static Logger getLogger(java.lang.Class clazz, java.lang.String group)</code></td>
<td>Shorthand for <code>getLogger(clazz.getName())</code>.</td>
</tr>
<tr>
<td><code>getLogger</code></td>
<td><code>static Logger getLogger(java.lang.String name)</code></td>
<td>Retrieve a logger named according to the value of the name parameter.</td>
</tr>
<tr>
<td><code>getLogger</code></td>
<td><code>static Logger getLogger(java.lang.String name, LoggerFactory factory)</code></td>
<td>Like <code>getLogger(String)</code> except that the type of logger instantiated depends on the type returned by the <code>LoggerFactory.makeNewLoggerInstance(java.lang.String)</code> method of the factory parameter.</td>
</tr>
<tr>
<td><code>getRootLogger</code></td>
<td><code>static Logger getRootLogger()</code></td>
<td>Return the root logger for the current logger repository.</td>
</tr>
<tr>
<td><code>info</code></td>
<td><code>void info(java.lang.String pattern, boolean argument)</code></td>
<td>Log a parameterized message at info level.</td>
</tr>
<tr>
<td><code>info</code></td>
<td><code>void info(java.lang.String pattern, byte argument)</code></td>
<td>Log a parameterized message at info level.</td>
</tr>
<tr>
<td><code>info</code></td>
<td><code>void info(java.lang.String pattern, char argument)</code></td>
<td>Log a parameterized message at info level.</td>
</tr>
<tr>
<td><code>info</code></td>
<td><code>void info(java.lang.String pattern, double argument)</code></td>
<td>Log a parameterized message at info level.</td>
</tr>
<tr>
<td><code>info</code></td>
<td><code>void info(java.lang.String pattern, float argument)</code></td>
<td>Log a parameterized message at info level.</td>
</tr>
</tbody>
</table>
## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void info(java.lang.String pattern, int argument)</code></td>
<td>Log a parameterized message at info level.</td>
</tr>
<tr>
<td><code>void info(java.lang.String pattern, long argument)</code></td>
<td>Log a parameterized message at info level.</td>
</tr>
<tr>
<td><code>void info(java.lang.String pattern, java.lang.Object[] arguments)</code></td>
<td>Log a parameterized message at info level.</td>
</tr>
<tr>
<td><code>void info(java.lang.String pattern, java.lang.Object arg0, java.lang.Object arg1)</code></td>
<td>Log a parameterized message at info level.</td>
</tr>
<tr>
<td><code>void info(java.lang.String pattern, java.lang.Throwable t)</code></td>
<td>Log a parameterized message at info level.</td>
</tr>
<tr>
<td><code>void info(java.lang.String pattern, short argument)</code></td>
<td>Log a parameterized message at info level.</td>
</tr>
<tr>
<td><code>void info(java.lang.Throwable t, java.lang.String pattern, java.lang.Object[] arguments)</code></td>
<td>Log a parameterized message at info level.</td>
</tr>
<tr>
<td><code>boolean isEnabled()</code></td>
<td>Check whether this category is enabled for the TRACE Level.</td>
</tr>
<tr>
<td><code>void log(Level level, java.lang.String pattern, boolean param1)</code></td>
<td>Log a parameterized message at specified level.</td>
</tr>
<tr>
<td><code>void log(Level level, java.lang.String pattern, byte param1)</code></td>
<td>Log a parameterized message at specified level.</td>
</tr>
<tr>
<td><code>void log(Level level, java.lang.String pattern, char param1)</code></td>
<td>Log a parameterized message at specified level.</td>
</tr>
<tr>
<td><code>void log(Level level, java.lang.String pattern, double param1)</code></td>
<td>Log a parameterized message at specified level.</td>
</tr>
<tr>
<td><code>void log(Level level, java.lang.String pattern, float param1)</code></td>
<td>Log a parameterized message at specified level.</td>
</tr>
<tr>
<td><code>void log(Level level, java.lang.String pattern, int param1)</code></td>
<td>Log a parameterized message at specified level.</td>
</tr>
<tr>
<td><code>void log(Level level, java.lang.String pattern, long param1)</code></td>
<td>Log a parameterized message at specified level.</td>
</tr>
<tr>
<td><code>void log(Level level, java.lang.String pattern, java.lang.Object param1)</code></td>
<td>Log a parameterized message at specified level.</td>
</tr>
</tbody>
</table>
## Method Summary

<table>
<thead>
<tr>
<th>Method Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`void log(Level level, java.lang.String pattern,</td>
<td>Log a parameterized message at specified level.</td>
</tr>
<tr>
<td>java.lang.Object[] parameters)`</td>
<td></td>
</tr>
<tr>
<td>`void log(Level level, java.lang.String pattern,</td>
<td>Log a parameterized message at specified level.</td>
</tr>
<tr>
<td>java.lang.Object arg0, java.lang.Object arg1)`</td>
<td></td>
</tr>
<tr>
<td>`void log(Level level, java.lang.String pattern,</td>
<td>Log a parameterized message at specified level.</td>
</tr>
<tr>
<td>java.lang.Object arg0, java.lang.Object arg1,</td>
<td></td>
</tr>
<tr>
<td>java.lang.Object arg2)`</td>
<td></td>
</tr>
<tr>
<td>`void log(Level level, java.lang.String pattern,</td>
<td>Log a parameterized message at specified level.</td>
</tr>
<tr>
<td>java.lang.Object arg0, java.lang.Object arg1,</td>
<td></td>
</tr>
<tr>
<td>java.lang.Object arg2, java.lang.Object arg3)`</td>
<td></td>
</tr>
<tr>
<td>`void log(Level level, java.lang.String pattern,</td>
<td>Log a parameterized message at specified level.</td>
</tr>
<tr>
<td>short param1)`</td>
<td></td>
</tr>
<tr>
<td>`void log(Level level, java.lang.Throwable t,</td>
<td>Log a parameterized message at specified level.</td>
</tr>
<tr>
<td>java.lang.String pattern, java.lang.Object[]</td>
<td></td>
</tr>
<tr>
<td>parameters)`</td>
<td></td>
</tr>
<tr>
<td>`void logrb(Level level, java.lang.String bundleName,</td>
<td>Log a parameterized message using a pattern from</td>
</tr>
<tr>
<td>java.lang.String key, boolean param1)`</td>
<td>a resource bundle.</td>
</tr>
<tr>
<td>`void logrb(Level level, java.lang.String bundleName,</td>
<td>Log a parameterized message using a pattern from</td>
</tr>
<tr>
<td>java.lang.String key, byte param1)`</td>
<td>a resource bundle.</td>
</tr>
<tr>
<td>`void logrb(Level level, java.lang.String bundleName,</td>
<td>Log a parameterized message using a pattern from</td>
</tr>
<tr>
<td>java.lang.String key, char param1)`</td>
<td>a resource bundle.</td>
</tr>
<tr>
<td>`void logrb(Level level, java.lang.String bundleName,</td>
<td>Log a parameterized message using a pattern from</td>
</tr>
<tr>
<td>java.lang.String key, double param1)`</td>
<td>a resource bundle.</td>
</tr>
<tr>
<td>`void logrb(Level level, java.lang.String bundleName,</td>
<td>Log a parameterized message using a pattern from</td>
</tr>
<tr>
<td>java.lang.String key, float param1)`</td>
<td>a resource bundle.</td>
</tr>
<tr>
<td>`void logrb(Level level, java.lang.String bundleName,</td>
<td>Log a parameterized message using a pattern from</td>
</tr>
<tr>
<td>java.lang.String key, int param1)`</td>
<td>a resource bundle.</td>
</tr>
<tr>
<td>`void logrb(Level level, java.lang.String bundleName,</td>
<td>Log a parameterized message using a pattern from</td>
</tr>
<tr>
<td>java.lang.String key, long param1)`</td>
<td>a resource bundle.</td>
</tr>
<tr>
<td>`void logrb(Level level, java.lang.String bundleName,</td>
<td>Log a parameterized message using a pattern from</td>
</tr>
<tr>
<td>`void logrb(Level level, java.lang.String bundleName,</td>
<td>Log a parameterized message using a pattern from</td>
</tr>
</tbody>
</table>
# Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>logrb</code></td>
<td>Log a parameterized message using a pattern from a resource bundle.</td>
</tr>
<tr>
<td><code>logrb</code></td>
<td>Log a parameterized message using a pattern from a resource bundle.</td>
</tr>
<tr>
<td><code>logrb</code></td>
<td>Log a parameterized message using a pattern from a resource bundle.</td>
</tr>
<tr>
<td><code>logrb</code></td>
<td>Log a parameterized message using a pattern from a resource bundle.</td>
</tr>
<tr>
<td><code>trace</code></td>
<td>Log a message object with the TRACE level.</td>
</tr>
<tr>
<td><code>trace</code></td>
<td>Log a message object with the TRACE level including the stack trace of the</td>
</tr>
<tr>
<td><code>trace</code></td>
<td>Log a parameterized message at trace level.</td>
</tr>
<tr>
<td><code>trace</code></td>
<td>Log a parameterized message at trace level.</td>
</tr>
<tr>
<td><code>trace</code></td>
<td>Log a parameterized message at trace level.</td>
</tr>
<tr>
<td><code>trace</code></td>
<td>Log a parameterized message at trace level.</td>
</tr>
<tr>
<td><code>trace</code></td>
<td>Log a parameterized message at trace level.</td>
</tr>
<tr>
<td><code>trace</code></td>
<td>Log a parameterized message at trace level.</td>
</tr>
<tr>
<td><code>trace</code></td>
<td>Log a parameterized message at trace level.</td>
</tr>
<tr>
<td><code>trace</code></td>
<td>Log a parameterized message at trace level.</td>
</tr>
</tbody>
</table>
### Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th><code>trace(java.lang.String pattern, java.lang.Object arg0, java.lang.Object arg1, java.lang.Object arg2)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Log a parameterized message at trace level.</td>
</tr>
<tr>
<td></td>
<td>Log a parameterized message at trace level.</td>
</tr>
<tr>
<td>void</td>
<td><code>trace(java.lang.String pattern, short argument)</code></td>
</tr>
<tr>
<td></td>
<td>Log a parameterized message at trace level.</td>
</tr>
<tr>
<td>void</td>
<td><code>trace(java.lang.Throwable t, java.lang.String pattern, java.lang.Object[] arguments)</code></td>
</tr>
<tr>
<td></td>
<td>Log a parameterized message at trace level.</td>
</tr>
<tr>
<td>void</td>
<td><code>warn(java.lang.String pattern, boolean argument)</code></td>
</tr>
<tr>
<td></td>
<td>Log a parameterized message at warn level.</td>
</tr>
<tr>
<td>void</td>
<td><code>warn(java.lang.String pattern, byte argument)</code></td>
</tr>
<tr>
<td></td>
<td>Log a parameterized message at warn level.</td>
</tr>
<tr>
<td>void</td>
<td><code>warn(java.lang.String pattern, char argument)</code></td>
</tr>
<tr>
<td></td>
<td>Log a parameterized message at warn level.</td>
</tr>
<tr>
<td>void</td>
<td><code>warn(java.lang.String pattern, double argument)</code></td>
</tr>
<tr>
<td></td>
<td>Log a parameterized message at warn level.</td>
</tr>
<tr>
<td>void</td>
<td><code>warn(java.lang.String pattern, float argument)</code></td>
</tr>
<tr>
<td></td>
<td>Log a parameterized message at warn level.</td>
</tr>
<tr>
<td>void</td>
<td><code>warn(java.lang.String pattern, int argument)</code></td>
</tr>
<tr>
<td></td>
<td>Log a parameterized message at warn level.</td>
</tr>
<tr>
<td>void</td>
<td><code>warn(java.lang.String pattern, long argument)</code></td>
</tr>
<tr>
<td></td>
<td>Log a parameterized message at warn level.</td>
</tr>
<tr>
<td>void</td>
<td><code>warn(java.lang.String pattern, java.lang.Object[] arguments)</code></td>
</tr>
<tr>
<td></td>
<td>Log a parameterized message at warn level.</td>
</tr>
<tr>
<td>void</td>
<td><code>warn(java.lang.String pattern, java.lang.Object arg0, java.lang.Object arg1)</code></td>
</tr>
<tr>
<td></td>
<td>Log a parameterized message at warn level.</td>
</tr>
<tr>
<td>void</td>
<td><code>warn(java.lang.String pattern, java.lang.Object arg0, java.lang.Object arg1, java.lang.Object arg2)</code></td>
</tr>
<tr>
<td></td>
<td>Log a parameterized message at warn level.</td>
</tr>
<tr>
<td></td>
<td>Log a parameterized message at warn level.</td>
</tr>
<tr>
<td>void</td>
<td><code>warn(java.lang.String pattern, java.lang.Object argument, java.lang.Throwable t)</code></td>
</tr>
<tr>
<td></td>
<td>Log a parameterized message at warn level.</td>
</tr>
<tr>
<td>void</td>
<td><code>warn(java.lang.String pattern, short argument)</code></td>
</tr>
<tr>
<td></td>
<td>Log a parameterized message at warn level.</td>
</tr>
</tbody>
</table>
Method Summary

```java
void warn(java.lang.Throwable t, java.lang.String pattern,
          java.lang.Object[] arguments)
Log a parameterized message at warn level.
```

Methods inherited from class org.apache.log4j.Category

- addAppender, assertLog, callAppenders, debug, debug, error, error, exists,
  fatal, fatal, getAdditivity, getAllAppenders, getAppend, getChainedPriority,
  getCurrentCategories, getDefau, getEffectiveLevel, getHierarchy,
  getInstances, getInstances, getLevel, getLogRepository, getName, getParent,
  getPriority, getResourceBundle, getRoot, info, info, isAttac
  isDebugEnabled, isEnabledFor, isErrorEnabled, isFatalEnabled, isInfoEnabled,
  isWarnEnabled, l7dlog, log, log, log, log, removeAppenders,
  removeAppender, removeAppender, setAdditivity, setLevel, setPriority,
  setResourceBundle, shutdown, warn, warn

Methods inherited from class java.lang.Object

- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString,
  wait, wait, wait

Constructor Detail

**Logger**

```java
public Logger()
```

Method Detail

**getLogger**

```java
public static Logger getLogger(java.lang.String name)
Retrieve a logger named according to the value of the name parameter. If the named logger already exists,
then the existing instance will be returned. Otherwise, a new instance is created.
```

Parameters:

- name - The name of the logger to retrieve.

**getLogger**

```java
public static Logger getLogger(java.lang.Class clazz)
Shorthand for getLogger(clazz.getName()).
```

Parameters:

- clazz - The name of clazz will be used as the name of the logger to retrieve. See
  getLogger(String) for more detailed information.

**getRootLogger**

```java
public static Logger getRootLogger()
Return the root logger for the current logger repository.
```
The Logger.getName() method for the root logger always returns string value: "root". However, calling Logger.getLogger("root") does not retrieve the root logger but a logger just under root named "root".

In other words, calling this method is the only way to retrieve the root logger.

getLogger

public static Logger getLogger(java.lang.String name,
                               LoggerFactory factory)

Like getLogger(String) except that the type of logger instantiated depends on the type returned by the LoggerFactory.makeNewLoggerInstance(java.lang.String) method of the factory parameter.

This method is intended to be used by sub-classes.

Parameters:
name - The name of the logger to retrieve.
factory - A LoggerFactory implementation that will actually create a new Instance.

Since: 0.8.5

getLogger

public static Logger getLogger(java.lang.String name,
                               java.lang.String group)

Retrieve a logger named according to the value of the name parameter. If the named logger already exists, then the existing instance will be returned. Otherwise, a new instance is created.

By default, loggers do not have a set level but inherit it from their nearest ancestor with a set level. This is one of the central features of log4j.

Parameters:
name - The name of the logger to retrieve.
group - The group this logger is associated with

getLogger

public static Logger getLogger(java.lang.Class clazz,
                               java.lang.String group)

Shorthand for getLogger(clazz.getName()).

Parameters:
class - The name of clazz will be used as the name of the logger to retrieve. See getLogger(String) for more detailed information.
group - The group this logger is associated with

getLogger

public static Logger getLogger(java.lang.String name,
                               java.lang.String group,
                               LoggerFactory factory)

Like getLogger(String) except that the type of logger instantiated depends on the type returned by the LoggerFactory.makeNewLoggerInstance(java.lang.String) method of the factory parameter.

This method is intended to be used by sub-classes.

Parameters:
name - The name of the logger to retrieve.
group - The group this logger is associated with
factory - A LoggerFactory implementation that will actually create a new Instance.
Since: 0.8.5

trace

public void trace(java.lang.Object message)
    Log a message object with the TRACE level.
Parameters:
message - the message object to log.
Since: 1.2.12
See Also: for an explanation of the logic applied.

trace

public void trace(java.lang.Object message,
                    java.lang.Throwable t)
    Log a message object with the TRACE level including the stack trace of the Throwable t passed as parameter.
    See Category.debug(Object) form for more detailed information.
Parameters:
message - the message object to log.
t - the exception to log, including its stack trace.
Since: 1.2.12

isTraceEnabled

public boolean isTraceEnabled()
    Check whether this category is enabled for the TRACE Level.
Returns:
boolean - true if this category is enabled for level TRACE, false otherwise.
Since: 1.2.12

trace

public void trace(java.lang.String pattern,
                    java.lang.Object[] arguments)
    Log a parameterized message at trace level.
Parameters:
pattern - pattern, may be null.
arguments - an array of arguments to be formatted and substituted.

debug

public void debug(java.lang.String pattern,
                    java.lang.Object[] arguments)
    Log a parameterized message at debug level.
Parameters:
pattern - pattern, may be null.
arguments - an array of arguments to be formatted and substituted.

info
public void info(java.lang.String pattern,
                 java.lang.Object[] arguments)
    Log a parameterized message at info level.
    Parameters:
    pattern - pattern, may be null.
    arguments - an array of arguments to be formatted and substituted.

warn
public void warn(java.lang.String pattern,
                  java.lang.Object[] arguments)
    Log a parameterized message at warn level.
    Parameters:
    pattern - pattern, may be null.
    arguments - an array of arguments to be formatted and substituted.

error
public void error(java.lang.String pattern,
                   java.lang.Object[] arguments)
    Log a parameterized message at error level.
    Parameters:
    pattern - pattern, may be null.
    arguments - an array of arguments to be formatted and substituted.

fatal
public void fatal(java.lang.String pattern,
                   java.lang.Object[] arguments)
    Log a parameterized message at fatal level.
    Parameters:
    pattern - pattern, may be null.
    arguments - an array of arguments to be formatted and substituted.

trace
public void trace(java.lang.Throwable t,
                   java.lang.String pattern,
                   java.lang.Object[] arguments)
    Log a parameterized message at trace level.
    Parameters:
    t - throwable, may be null.
    pattern - pattern, may be null.
    arguments - an array of arguments to be formatted and substituted.

debug
public void debug(java.lang.Throwable t,
                   java.lang.String pattern,
                   java.lang.Object[] arguments)
Log a parameterized message at debug level.
Parameters:
t - throwable, may be null.
pattern - pattern, may be null.
arguments - an array of arguments to be formatted and substituted.

info
public void info(java.lang.Throwable t,
        java.lang.String pattern,
        java.lang.Object[] arguments)
Log a parameterized message at info level.
Parameters:
t - throwable, may be null.
pattern - pattern, may be null.
arguments - an array of arguments to be formatted and substituted.

warn
public void warn(java.lang.Throwable t,
        java.lang.String pattern,
        java.lang.Object[] arguments)
Log a parameterized message at warn level.
Parameters:
t - throwable, may be null.
pattern - pattern, may be null.
arguments - an array of arguments to be formatted and substituted.

error
public void error(java.lang.Throwable t,
        java.lang.String pattern,
        java.lang.Object[] arguments)
Log a parameterized message at error level.
Parameters:
t - throwable, may be null.
pattern - pattern, may be null.
arguments - an array of arguments to be formatted and substituted.

fatal
public void fatal(java.lang.Throwable t,
        java.lang.String pattern,
        java.lang.Object[] arguments)
Log a parameterized message at fatal level.
Parameters:
t - throwable, may be null.
pattern - pattern, may be null.
arguments - an array of arguments to be formatted and substituted.

trace
public void trace(java.lang.String pattern,
        boolean argument)
Log a parameterized message at trace level.
Parameters:
- pattern - pattern, may be null.
- argument - a value to be formatted and substituted.

trace
public void trace(java.lang.String pattern,
                   char argument)
    Log a parameterized message at trace level.
Parameters:
- pattern - pattern, may be null.
- argument - a value to be formatted and substituted.

trace
public void trace(java.lang.String pattern,
                   byte argument)
    Log a parameterized message at trace level.
Parameters:
- pattern - pattern, may be null.
- argument - a value to be formatted and substituted.

trace
public void trace(java.lang.String pattern,
                   short argument)
    Log a parameterized message at trace level.
Parameters:
- pattern - pattern, may be null.
- argument - a value to be formatted and substituted.

trace
public void trace(java.lang.String pattern,
                   int argument)
    Log a parameterized message at trace level.
Parameters:
- pattern - pattern, may be null.
- argument - a value to be formatted and substituted.

trace
public void trace(java.lang.String pattern,
                   long argument)
    Log a parameterized message at trace level.
Parameters:
- pattern - pattern, may be null.
- argument - a value to be formatted and substituted.

trace
public void trace(java.lang.String pattern,
                   float argument)
    Log a parameterized message at trace level.
Parameters:
pattern - pattern, may be null.
argument - a value to be formatted and substituted.

```java
public void trace(java.lang.String pattern,
                  double argument)
```
Log a parameterized message at trace level.

Parameters:
- pattern - pattern, may be null.
- argument - a value to be formatted and substituted.

```java
public void trace(java.lang.String pattern,
                  java.lang.Object argument)
```
Log a parameterized message at trace level.

Parameters:
- pattern - pattern, may be null.
- argument - a value to be formatted and substituted.

```java
public void trace(java.lang.String pattern,
                  java.lang.Object arg0,
                  java.lang.Object arg1)
```
Log a parameterized message at trace level.

Parameters:
- pattern - pattern, may be null.
- arg0 - a value to be formatted and substituted.
- arg1 - a value to be formatted and substituted.

```java
public void trace(java.lang.String pattern,
                  java.lang.Object arg0,
                  java.lang.Object arg1,
                  java.lang.Object arg2)
```
Log a parameterized message at trace level.

Parameters:
- pattern - pattern, may be null.
- arg0 - a value to be formatted and substituted.
- arg1 - a value to be formatted and substituted.
- arg2 - a value to be formatted and substituted.

```java
public void trace(java.lang.String pattern,
                  java.lang.Object arg0,
                  java.lang.Object arg1,
                  java.lang.Object arg2,
                  java.lang.Object arg3)
```
Log a parameterized message at trace level.

Parameters:
- pattern - pattern, may be null.
arg0 - a value to be formatted and substituted.
arg1 - a value to be formatted and substituted.
arg2 - a value to be formatted and substituted.
arg3 - a value to be formatted and substituted.

**debug**

```java
public void debug(java.lang.String pattern,
                  boolean argument)
```

Log a parameterized message at debug level.

**Parameters:**
- pattern - pattern, may be null.
- argument - a value to be formatted and substituted.

**debug**

```java
public void debug(java.lang.String pattern,
                  char argument)
```

Log a parameterized message at debug level.

**Parameters:**
- pattern - pattern, may be null.
- argument - a value to be formatted and substituted.

**debug**

```java
public void debug(java.lang.String pattern,
                  byte argument)
```

Log a parameterized message at debug level.

**Parameters:**
- pattern - pattern, may be null.
- argument - a value to be formatted and substituted.

**debug**

```java
public void debug(java.lang.String pattern,
                  short argument)
```

Log a parameterized message at debug level.

**Parameters:**
- pattern - pattern, may be null.
- argument - a value to be formatted and substituted.

**debug**

```java
public void debug(java.lang.String pattern,
                  int argument)
```

Log a parameterized message at debug level.

**Parameters:**
- pattern - pattern, may be null.
- argument - a value to be formatted and substituted.

**debug**

```java
public void debug(java.lang.String pattern,
                  long argument)
```

Log a parameterized message at debug level.
Parameters:
pattern - pattern, may be null.
argument - a value to be formatted and substituted.

```java
public void debug(java.lang.String pattern,
                  float argument)
```
Log a parameterized message at debug level.

Parameters:
pattern - pattern, may be null.
argument - a value to be formatted and substituted.

```java
public void debug(java.lang.String pattern,
                  double argument)
```
Log a parameterized message at debug level.

Parameters:
pattern - pattern, may be null.
argument - a value to be formatted and substituted.

```java
public void debug(java.lang.String pattern,
                  java.lang.Object argument,
                  java.lang.Throwable t)
```
Log a parameterized message at debug level.

Parameters:
pattern - pattern, may be null.
argument - a value to be formatted and substituted.
t - a throwable, may be null.

```java
public void debug(java.lang.String pattern,
                  java.lang.Object arg0,
                  java.lang.Object arg1)
```
Log a parameterized message at debug level.

Parameters:
pattern - pattern, may be null.
arg0 - a value to be formatted and substituted.
arg1 - a value to be formatted and substituted.

```java
public void debug(java.lang.String pattern,
                  java.lang.Object arg0,
                  java.lang.Object arg1,
                  java.lang.Object arg2)
```
Log a parameterized message at debug level.

Parameters:
pattern - pattern, may be null.
arg0 - a value to be formatted and substituted.
arg1 - a value to be formatted and substituted.
arg2 - a value to be formatted and substituted.

debug

public void debug(java.lang.String pattern,
                 java.lang.Object arg0,
                 java.lang.Object arg1,
                 java.lang.Object arg2,
                 java.lang.Object arg3)

Log a parameterized message at debug level.
Parameters:
pattern - pattern, may be null.
arg0 - a value to be formatted and substituted.
arg1 - a value to be formatted and substituted.
arg2 - a value to be formatted and substituted.
arg3 - a value to be formatted and substituted.

info

public void info(java.lang.String pattern,
                 boolean argument)

Log a parameterized message at info level.
Parameters:
pattern - pattern, may be null.
argument - a value to be formatted and substituted.

info

public void info(java.lang.String pattern,
                 char argument)

Log a parameterized message at info level.
Parameters:
pattern - pattern, may be null.
argument - a value to be formatted and substituted.

info

public void info(java.lang.String pattern,
                 byte argument)

Log a parameterized message at info level.
Parameters:
pattern - pattern, may be null.
argument - a value to be formatted and substituted.

info

public void info(java.lang.String pattern,
                 short argument)

Log a parameterized message at info level.
Parameters:
pattern - pattern, may be null.
argument - a value to be formatted and substituted.
public void info(java.lang.String pattern, int argument)
    Log a parameterized message at info level.
    Parameters:
    pattern - pattern, may be null.
    argument - a value to be formatted and substituted.

public void info(java.lang.String pattern, long argument)
    Log a parameterized message at info level.
    Parameters:
    pattern - pattern, may be null.
    argument - a value to be formatted and substituted.

public void info(java.lang.String pattern, float argument)
    Log a parameterized message at info level.
    Parameters:
    pattern - pattern, may be null.
    argument - a value to be formatted and substituted.

public void info(java.lang.String pattern, double argument)
    Log a parameterized message at info level.
    Parameters:
    pattern - pattern, may be null.
    argument - a value to be formatted and substituted.

public void info(java.lang.String pattern, java.lang.Object argument, java.lang.Throwable t)
    Log a parameterized message at info level.
    Parameters:
    pattern - pattern, may be null.
    argument - a value to be formatted and substituted.
    t - a throwable, may be null.

public void info(java.lang.String pattern, java.lang.Object arg0, java.lang.Object arg1)
    Log a parameterized message at info level.
    Parameters:
    pattern - pattern, may be null.
    arg0 - a value to be formatted and substituted.
arg1 - a value to be formatted and substituted.

info

public void info(java.lang.String pattern,
    java.lang.Object arg0,
    java.lang.Object arg1,
    java.lang.Object arg2)

Log a parameterized message at info level.
Parameters:
pattern - pattern, may be null.
arg0 - a value to be formatted and substituted.
arg1 - a value to be formatted and substituted.
arg2 - a value to be formatted and substituted.

info

public void info(java.lang.String pattern,
    java.lang.Object arg0,
    java.lang.Object arg1,
    java.lang.Object arg2,
    java.lang.Object arg3)

Log a parameterized message at info level.
Parameters:
pattern - pattern, may be null.
arg0 - a value to be formatted and substituted.
arg1 - a value to be formatted and substituted.
arg2 - a value to be formatted and substituted.
arg3 - a value to be formatted and substituted.

warn

public void warn(java.lang.String pattern,
    boolean argument)

Log a parameterized message at warn level.
Parameters:
pattern - pattern, may be null.
argument - a value to be formatted and substituted.

warn

public void warn(java.lang.String pattern,
    char argument)

Log a parameterized message at warn level.
Parameters:
pattern - pattern, may be null.
argument - a value to be formatted and substituted.

warn

public void warn(java.lang.String pattern,
    byte argument)

Log a parameterized message at warn level.
Parameters:
pattern - pattern, may be null.
argument - a value to be formatted and substituted.

**warn**

public void **warn**(java.lang.String pattern,
short argument)

Log a parameterized message at warn level.

**Parameters:**

pattern - pattern, may be null.
argument - a value to be formatted and substituted.

**warn**

public void **warn**(java.lang.String pattern,
int argument)

Log a parameterized message at warn level.

**Parameters:**

pattern - pattern, may be null.
argument - a value to be formatted and substituted.

**warn**

public void **warn**(java.lang.String pattern,
long argument)

Log a parameterized message at warn level.

**Parameters:**

pattern - pattern, may be null.
argument - a value to be formatted and substituted.

**warn**

public void **warn**(java.lang.String pattern,
float argument)

Log a parameterized message at warn level.

**Parameters:**

pattern - pattern, may be null.
argument - a value to be formatted and substituted.

**warn**

public void **warn**(java.lang.String pattern,
double argument)

Log a parameterized message at warn level.

**Parameters:**

pattern - pattern, may be null.
argument - a value to be formatted and substituted.

**warn**

public void **warn**(java.lang.String pattern,
java.lang.Object argument,
java.lang.Throwable t)

Log a parameterized message at warn level.

**Parameters:**

pattern - pattern, may be null.
argument - a value to be formatted and substituted.  
t - a throwable, may be null.

```java
public void warn(java.lang.String pattern,
                 java.lang.Object arg0,
                 java.lang.Object arg1)

Log a parameterized message at warn level.
Parameters:
  pattern - pattern, may be null.
  arg0 - a value to be formatted and substituted.
  arg1 - a value to be formatted and substituted.
```

```java
public void warn(java.lang.String pattern,
                 java.lang.Object arg0,
                 java.lang.Object arg1,
                 java.lang.Object arg2)

Log a parameterized message at warn level.
Parameters:
  pattern - pattern, may be null.
  arg0 - a value to be formatted and substituted.
  arg1 - a value to be formatted and substituted.
  arg2 - a value to be formatted and substituted.
```

```java
public void warn(java.lang.String pattern,
                 java.lang.Object arg0,
                 java.lang.Object arg1,
                 java.lang.Object arg2,
                 java.lang.Object arg3)

Log a parameterized message at warn level.
Parameters:
  pattern - pattern, may be null.
  arg0 - a value to be formatted and substituted.
  arg1 - a value to be formatted and substituted.
  arg2 - a value to be formatted and substituted.
  arg3 - a value to be formatted and substituted.
```

```java
public void log(Level level,
                java.lang.String pattern,
                java.lang.Object[] parameters)

Log a parameterized message at specified level.
Parameters:
  level - level, may not be null.
  pattern - pattern, may be null.
  parameters - parameters to the log message.
```
public void log(Level level,
     java.lang.Throwable t,
     java.lang.String pattern,
     java.lang.Object[] parameters)

    Log a parameterized message at specified level.
    Parameters:
    level - level, may not be null.
    t - throwable, may be null.
    pattern - pattern, may be null.
    parameters - parameters to the log message.

public void log(Level level,
     java.lang.String pattern,
     java.lang.Object param1)

    Log a parameterized message at specified level.
    Parameters:
    level - level, may not be null.
    pattern - pattern, may be null.
    param1 - parameter to the log message.

public void log(Level level,
     java.lang.String pattern,
     boolean param1)

    Log a parameterized message at specified level.
    Parameters:
    level - level, may not be null.
    pattern - pattern, may be null.
    param1 - parameter to the log message.

public void log(Level level,
     java.lang.String pattern,
     byte param1)

    Log a parameterized message at specified level.
    Parameters:
    level - level, may not be null.
    pattern - pattern, may be null.
    param1 - parameter to the log message.

public void log(Level level,
     java.lang.String pattern,
     char param1)

    Log a parameterized message at specified level.
    Parameters:
    level - level, may not be null.
    pattern - pattern, may be null.
    param1 - parameter to the log message.
public void log(Level level, java.lang.String pattern, short param1)
Log a parameterized message at specified level.
Parameters:
level - level, may not be null.
pattern - pattern, may be null.
param1 - parameter to the log message.

public void log(Level level, java.lang.String pattern, int param1)
Log a parameterized message at specified level.
Parameters:
level - level, may not be null.
pattern - pattern, may be null.
param1 - parameter to the log message.

public void log(Level level, java.lang.String pattern, long param1)
Log a parameterized message at specified level.
Parameters:
level - level, may not be null.
pattern - pattern, may be null.
param1 - parameter to the log message.

public void log(Level level, java.lang.String pattern, float param1)
Log a parameterized message at specified level.
Parameters:
level - level, may not be null.
pattern - pattern, may be null.
param1 - parameter to the log message.

public void log(Level level, java.lang.String pattern, double param1)
Log a parameterized message at specified level.
Parameters:
level - level, may not be null.
pattern - pattern, may be null.
param1 - parameter to the log message.
log
public void log(Level level,
        java.lang.String pattern,
        java.lang.Object arg0,
        java.lang.Object arg1)
    Log a parameterized message at specified level.
    Parameters:
    level - level, may not be null.
    pattern - pattern, may be null.
    arg0 - a value to be formatted and substituted.
    arg1 - a value to be formatted and substituted.

log
public void log(Level level,
        java.lang.String pattern,
        java.lang.Object arg0,
        java.lang.Object arg1,
        java.lang.Object arg2)
    Log a parameterized message at specified level.
    Parameters:
    level - level, may not be null.
    pattern - pattern, may be null.
    arg0 - a value to be formatted and substituted.
    arg1 - a value to be formatted and substituted.
    arg2 - a value to be formatted and substituted.

log
public void log(Level level,
        java.lang.String pattern,
        java.lang.Object arg0,
        java.lang.Object arg1,
        java.lang.Object arg2,
        java.lang.Object arg3)
    Log a parameterized message at specified level.
    Parameters:
    pattern - pattern, may be null.
    level - level, may not be null.
    arg0 - a value to be formatted and substituted.
    arg1 - a value to be formatted and substituted.
    arg2 - a value to be formatted and substituted.
    arg3 - a value to be formatted and substituted.

logrb
public void logrb(Level level,
        java.lang.String bundleName,
        java.lang.String key,
        java.lang.Object[] parameters)
    Log a parameterized message using a pattern from a resource bundle.
    Parameters:
    level - level, may not be null.
    bundleName - resource bundle name, may be null.
key - key, may be null.
parameters - parameters to the log message.

logrb

public void logrb(Level level,
java.lang.Throwable t,
java.lang.String bundleName,
java.lang.String key,
java.lang.Object[] parameters)

Log a parameterized message using a pattern from a resource bundle.

Parameters:
level - level, may not be null.
t - throwable, may be null.
bundleName - resource bundle name, may be null.
key - key, may be null.
parameters - parameters to the log message.

logrb

public void logrb(Level level,
java.lang.String bundleName,
java.lang.String key,
java.lang.Object param1)

Log a parameterized message using a pattern from a resource bundle.

Parameters:
level - level, may not be null.
bundleName - resource bundle name, may be null.
key - key, may be null.
param1 - Parameter to the log message.

logrb

public void logrb(Level level,
java.lang.String bundleName,
java.lang.String key,
boolean param1)

Log a parameterized message using a pattern from a resource bundle.

Parameters:
level - level, may not be null.
bundleName - resource bundle name, may be null.
key - key, may be null.
param1 - Parameter to the log message.

logrb

public void logrb(Level level,
java.lang.String bundleName,
java.lang.String key,
char param1)

Log a parameterized message using a pattern from a resource bundle.

Parameters:
level - level, may not be null.
bundleName - resource bundle name, may be null.
key - key, may be null.
param1 - Parameter to the log message.
Log a parameterized message using a pattern from a resource bundle.

**Parameters:**
- `level` - level, may not be null.
- `bundleName` - resource bundle name, may be null.
- `key` - key, may be null.
- `param1` - Parameter to the log message.

Log a parameterized message using a pattern from a resource bundle.

**Parameters:**
- `level` - level, may not be null.
- `bundleName` - resource bundle name, may be null.
- `key` - key, may be null.
- `param1` - Parameter to the log message.

Log a parameterized message using a pattern from a resource bundle.

**Parameters:**
- `level` - level, may not be null.
- `bundleName` - resource bundle name, may be null.
- `key` - key, may be null.
- `param1` - Parameter to the log message.

Log a parameterized message using a pattern from a resource bundle.

**Parameters:**
- `level` - level, may not be null.
- `bundleName` - resource bundle name, may be null.
- `key` - key, may be null.
- `param1` - Parameter to the log message.
java.lang.String bundleName,
java.lang.String key,
float param1)

Log a parameterized message using a pattern from a resource bundle.

**Parameters:**
- level - level, may not be null.
- bundleName - resource bundle name, may be null.
- key - key, may be null.
- param1 - Parameter to the log message.

logrb

```java
public void logrb(Level level,
                   java.lang.String bundleName,
                   java.lang.String key,
                   double param1)
```

Log a parameterized message using a pattern from a resource bundle.

**Parameters:**
- level - level, may not be null.
- bundleName - resource bundle name, may be null.
- key - key, may be null.
- param1 - Parameter to the log message.

logrb

```java
public void logrb(Level level,
                   java.lang.String bundleName,
                   java.lang.String key,
                   java.lang.Object param0,
                   java.lang.Object param1)
```

Log a parameterized message using a pattern from a resource bundle.

**Parameters:**
- level - level, may not be null.
- bundleName - resource bundle name, may be null.
- key - key, may be null.
- param0 - Parameter to the log message.
- param1 - Parameter to the log message.

logrb

```java
public void logrb(Level level,
                   java.lang.String bundleName,
                   java.lang.String key,
                   java.lang.Object param0,
                   java.lang.Object param1,
                   java.lang.Object param2)
```

Log a parameterized message using a pattern from a resource bundle.

**Parameters:**
- level - level, may not be null.
- bundleName - resource bundle name, may be null.
- key - key, may be null.
- param0 - Parameter to the log message.
- param1 - Parameter to the log message.
- param2 - Parameter to the log message.
public void logrb(Level level,
        java.lang.String bundleName,
        java.lang.String key,
        java.lang.Object param0,
        java.lang.Object param1,
        java.lang.Object param2,
        java.lang.Object param3)

Log a parameterized message using a pattern from a resource bundle.

Parameters:
level - level, may not be null.
bundleName - resource bundle name, may be null.
key - key, may be null.
param0 - Parameter to the log message.
param1 - Parameter to the log message.
param2 - Parameter to the log message.
param3 - Parameter to the log message.
org.apache.log4j

Class LogManager

java.lang.Object
   \org.apache.log4j.LogManager

public class LogManager
   extends java.lang.Object

Use the LogManager class to retrieve Logger instances or to operate on the current LoggerRepository. When the LogManager class is loaded into memory the default initialization procedure is initiated. The default initialization procedure is described in the short log4j manual.

Constructor Summary

| LogManager() |

Method Summary

| static Logger exists(java.lang.String name) |
| static java.util.Enumeration getCurrentLoggers() |
| static Logger getLogger(java.lang.Class clazz) |
| static Logger getLogger(java.lang.String name) |
| static Logger getLogger(java.lang.String name, LoggerFactory factory) |
| static LoggerRepository getLoggerRepository() |
| static Logger getRootLogger() |
| static void resetConfiguration() |
| static void setRepositorySelector(RepositorySelector selector, java.lang.Object guard) |
| static void shutdown() |
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

LogManager
public LogManager()

Method Detail

setRepositorySelector
public static void setRepositorySelector(RepositorySelector selector, java.lang.Object guard) throws java.lang.IllegalArgumentException
Sets LoggerFactory but only if the correct guard is passed as parameter.
Initially the guard is null. If the guard is null, then invoking this method sets the logger factory and the guard. Following invocations will throw a IllegalArgumentException, unless the previously set guard is passed as the second parameter.

This allows a high-level component to set the RepositorySelector used by the LogManager.

For example, when tomcat starts it will be able to install its own repository selector. However, if and when Tomcat is embedded within JBoss, then JBoss will install its own repository selector and Tomcat will use the repository selector set by its container, JBoss.

Throws:
java.lang.IllegalArgumentException

getLoggerRepository
public static LoggerRepository getLoggerRepository()

getRootLogger
public static Logger getRootLogger()
Retrieve the appropriate root logger.

getLogger
public static Logger getLogger(java.lang.String name)
Retrieve the appropriate Logger instance.

getLogger
public static Logger getLogger(java.lang.Class clazz)
Retrieve the appropriate Logger instance.

getLogger
public static Logger getLogger(java.lang.String name, LoggerFactory factory)
Retrieve the appropriate Logger instance.

exists
public static Logger exists(java.lang.String name)

getCurrentLoggers
public static java.util.Enumeration getCurrentLoggers()

shutdown
public static void shutdown()

resetConfiguration
public static void resetConfiguration()
org.apache.log4j
Class PropertyConfigurator

`java.lang.Object`
   `org.apache.log4j.PropertyConfigurator`

All Implemented Interfaces:
   Configurator

public class PropertyConfigurator
extends java.lang.Object
implements Configurator

Allows the configuration of log4j from an external file. See `doConfigure(String, LoggerRepository)` for the expected format.

It is sometimes useful to see how log4j is reading configuration files. You can enable log4j internal logging by defining the `log4j.debug` variable.

As of log4j version 0.8.5, at class initialization time class, the file `log4j.properties` will be searched from the search path used to load classes. If the file can be found, then it will be fed to the `configure(java.net.URL)` method.

The PropertyConfigurator does not handle the advanced configuration features supported by the DOMConfigurator such as support custom ErrorHandlers, nested appenders such as the AsyncAppender, etc. This version of PropertyConfigurator has been modified to support the definition of groups as described in Logger by adding a 'group' entry using this format:

log4j.group.NAME=LEVEL

All loggers retrieved via `Logger.getLogger(String, String)` or `Logger.getLogger(Class, String)` with the same group name will have their effective level set to the provided level.

This group-defined Level verbosity threshold can be overridden by setting an explicit level on the Logger itself.

All option values admit variable substitution. The syntax of variable substitution is similar to that of Unix shells. The string between an opening "${" and closing "}" is interpreted as a key. The value of the substituted variable can be defined as a system property or in the configuration file itself. The value of the key is first searched in the system properties, and if not found there, it is then searched in the configuration file being parsed. The corresponding value replaces the `${variableName}` sequence. For example, if `java.home` system property is set to `/home/xyz`, then every occurrence of the sequence `${java.home}` will be interpreted as `/home/xyz`.

Since:
   0.8.1

Field Summary

Fields inherited from interface org.apache.log4j.spi.Configurator

| INHERITED, NULL |
Constructor Summary

PropertyConfigurator()

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void configure(java.util.Properties properties)</td>
<td>Read configuration options from properties.</td>
</tr>
<tr>
<td>static void configure(java.lang.String configFilename)</td>
<td>Read configuration options from file configFilename.</td>
</tr>
<tr>
<td>static void configure(java.net.URL configURL)</td>
<td>Read configuration options from url configURL.</td>
</tr>
<tr>
<td>static void configureAndWatch(java.lang.String configFilename)</td>
<td>Like configureAndWatch(String, long) except that the default delay as defined by FileWatchdog#DEFAULT_DELAY is used.</td>
</tr>
<tr>
<td>static void configureAndWatch(java.lang.String configFilename, long delay)</td>
<td>Read the configuration file configFilename if it exists.</td>
</tr>
<tr>
<td>void doConfigure(java.util.Properties properties, LoggerRepository hierarchy)</td>
<td>Read configuration options from properties.</td>
</tr>
<tr>
<td>void doConfigure(java.lang.String configFileName, LoggerRepository hierarchy)</td>
<td>Read configuration from a file.</td>
</tr>
<tr>
<td>void doConfigure(java.net.URL configURL, LoggerRepository hierarchy)</td>
<td>Read configuration options from url configURL.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

PropertyConfigurator
public PropertyConfigurator()

Method Detail

doConfigure
public void doConfigure(java.lang.String configFileName, LoggerRepository hierarchy)

Read configuration from a file. The existing configuration is not cleared nor reset. If you require a different behavior, then call resetConfiguration method before calling doConfigure.

The configuration file consists of statements in the format key=value. The syntax of different configuration elements are discussed below.
Repository-wide threshold

The repository-wide threshold filters logging requests by level regardless of logger. The syntax is:

```properties
log4j.threshold=[level]
```

The level value can consist of the string values OFF, FATAL, ERROR, WARN, INFO, DEBUG, ALL or a custom level value. A custom level value can be specified in the form `level#classname`. By default the repository-wide threshold is set to the lowest possible value, namely the level ALL.

Appender configuration

Appender configuration syntax is:

```properties
# For appender named appenderName, set its class.
# Note: The appender name can contain dots.
log4j.appender.appenderName=fully.qualified.name.of.appender.class

# Set appender specific options.
log4j.appender.appenderName.option1=value1
... 
log4j.appender.appenderName.optionN=valueN
```

For each named appender you can configure its Layout. The syntax for configuring an appender's layout is:

```properties
log4j.appender.appenderName.layout=fully.qualified.name.of.layout.class
log4j.appender.appenderName.layout.option1=value1
... 
log4j.appender.appenderName.layout.optionN=valueN
```

The syntax for adding Filters to an appender is:

```properties
log4j.appender.appenderName.filter.ID=fully.qualified.name.of.filter.class
log4j.appender.appenderName.filter.ID.option1=value1
... 
log4j.appender.appenderName.filter.ID.optionN=valueN
```

The first line defines the class name of the filter identified by ID; subsequent lines with the same ID specify filter option - value pairs. Multiple filters are added to the appender in the lexicographic order of IDs. The syntax for adding an ErrorHandler to an appender is:

```properties
log4j.appender.appenderName.errorhandler=fully.qualified.name.of.filter.class
log4j.appender.appenderName.errorhandler.root-ref={true|false}
log4j.appender.appenderName.errorhandler.logger-ref=loggerName
log4j.appender.appenderName.errorhandler.appender-ref=appenderName
log4j.appender.appenderName.errorhandler.option1=value1
... 
log4j.appender.appenderName.errorhandler.optionN=valueN
```
Configuring loggers

The syntax for configuring the root logger is:

    log4j.rootLogger=[level], appenderName, appenderName, ...

This syntax means that an optional level can be supplied followed by appender names separated by commas.

The level value can consist of the string values OFF, FATAL, ERROR, WARN, INFO, DEBUG, ALL or a custom level value. A custom level value can be specified in the form level#classname.

If a level value is specified, then the root level is set to the corresponding level. If no level value is specified, then the root level remains untouched.

The root logger can be assigned multiple appenders.

Each appenderName (separated by commas) will be added to the root logger. The named appender is defined using the appender syntax defined above.

For non-root categories the syntax is almost the same:

    log4j.logger.logger_name=[level|INHERITED|NULL], appenderName, appenderName, ...

The meaning of the optional level value is discussed above in relation to the root logger. In addition however, the value INHERITED can be specified meaning that the named logger should inherit its level from the logger hierarchy.

If no level value is supplied, then the level of the named logger remains untouched.

By default categories inherit their level from the hierarchy. However, if you set the level of a logger and later decide that that logger should inherit its level, then you should specify INHERITED as the value for the level value. NULL is a synonym for INHERITED.

Similar to the root logger syntax, each appenderName (separated by commas) will be attached to the named logger.

See the appender additivity rule in the user manual for the meaning of the additivity flag.

ObjectRenderers

You can customize the way message objects of a given type are converted to String before being logged. This is done by specifying an ObjectRenderer for the object type would like to customize.

The syntax is:

    log4j.renderer.fully.qualified.name.of.rendered.class=fully.qualified.name.of.rendering.class

As in,

    log4j.renderer.my.Fruit=my.FruitRenderer
ThrowableRenderer

You can customize the way an instance of Throwable is converted to String before being logged. This is done by specifying a ThrowableRenderer.

The syntax is:

```
log4j.throwableRenderer=fully.qualified.name.of.rendering.class
log4j.throwableRenderer.paramName=paramValue
```

As in,

```
log4j.throwableRenderer=org.apache.log4j.EnhancedThrowableRenderer
```

Logger Factories

The usage of custom logger factories is discouraged and no longer documented.

Resetting Hierarchy

The hierarchy will be reset before configuration when log4j.reset=true is present in the properties file.

Example

An example configuration is given below. Other configuration file examples are given in the examples folder.

```
# Set options for appender named "A1".
# Appender "A1" will be a SyslogAppender
log4j.appender.A1=org.apache.log4j.net.SyslogAppender

# The syslog daemon resides on www.abc.net

# A1's layout is a PatternLayout, using the conversion pattern
# %r %-5p %c{2} %M.%L %x - %m\n. Thus, the log output will
# include # the relative time since the start of the application in
# milliseconds, followed by the level of the log request,
# followed by the two rightmost components of the logger name,
# followed by the callers method name, followed by the line number,
# the nested diagnostic context and finally the message itself.
# Refer to the documentation of PatternLayout for further
# information
# on the syntax of the ConversionPattern key.
log4j.appender.A1.layout.ConversionPattern=%-4r %-5p %c{2} %M.%L %x - %m\n
# Set options for appender named "A2"
# A2 should be a RollingFileAppender, with maximum file size of 10 MB
# using at most one backup file. A2's layout is TTCC, using the
# ISO8601 date format with context printing enabled.
log4j.appender.A2=org.apache.log4j.RollingFileAppender
log4j.appender.A2.MaxFileSize=10MB
log4j.appender.A2.MaxBackupIndex=1
log4j.appender.A2.layout=org.apache.log4j.TTCCLayout
log4j.appender.A2.layout.ContextPrinting=enabled
log4j.appender.A2.layout.DateFormat=ISO8601
```
# Root logger set to DEBUG using the A2 appender defined above.
log4j.rootLogger=DEBUG, A2

# Logger definitions:
# The SECURITY logger inherits its level from root. However, it's output
# will go to A1 appender defined above. It's additivity is non-cumulative.
log4j.logger.SECURITY=INHERIT, A1
log4j.additivity.SECURITY=false

# Only warnings or above will be logged for the logger "SECURITY.access".
# Output will go to A1.
log4j.logger.SECURITY.access=WARN

# The logger "class.of.the.day" inherits its level from the
# logger hierarchy. Output will go to the appender's of the root
# logger, A2 in this case.
log4j.logger.class.of.the.day=INHERIT

Refer to the setOption method in each Appender and Layout for class specific options.

Use the # or ! characters at the beginning of a line for comments.

Parameters:
configFileName - The name of the configuration file where the configuration information is stored.

Throws:
java.lang.SecurityException - if the caller does not have MonitorAppPermission("logger.config").

configure

class configure(java.lang.String configFilename)

Throws:
java.lang.SecurityException - if the caller does not have MonitorAppPermission("logger.config").

configure

class configure(java.net.URL configURL)

Read configuration options from url configURL.

Throws:
java.lang.SecurityException - if the caller does not have MonitorAppPermission("logger.config").

Since:
0.8.2

configure

class configure(java.util.Properties properties)

Read configuration options from properties. See doConfigure(String, LoggerRepository) for the expected format.

Throws:
java.lang.SecurityException - if the caller does not have
MonitorAppPermission("logger.config").

configureAndWatch

public static void configureAndWatch(java.lang.String configFilename)
Like configureAndWatch(String, long) except that the default delay as defined by
FileWatchdog#DEFAULT_DELAY is used.
Parameters:
configFilename - A file in key=value format.
Throws:
java.lang.SecurityException - if the caller does not have
MonitorAppPermission("logger.config").

class com.cablelabs.oap.app.framework.config.FileWatcher

public static void configureAndWatch(java.lang.String configFilename,
long delay)
Read the configuration file configFilename if it exists. Moreover, a thread will be created that will
periodically check if configFilename has been created or modified. The period is determined by the
delay argument. If a change or file creation is detected, then configFilename is read to configure
log4j.
Parameters:
configFilename - A file in key=value format.
delay - The delay in milliseconds to wait between each check.
Throws:
java.lang.SecurityException - if the caller does not have
MonitorAppPermission("logger.config").

doConfigure

public void doConfigure(java.util.Properties properties,
LoggerRepository hierarchy)
Read configuration options from properties. See doConfigure(String,
LoggerRepository) for the expected format.
Throws:
java.lang.SecurityException - if the caller does not have
MonitorAppPermission("logger.config").

doConfigure

public void doConfigure(java.net.URL configURL,
LoggerRepository hierarchy)
Read configuration options from url configURL.
Specified by:
doConfigure in interface Configurator
Parameters:
cfgURL - The URL to parse
hierarchy - The hierarchy to operation upon.
Throws:
java.lang.SecurityException - if the caller does not have
MonitorAppPermission("logger.config").
Package org.apache.log4j.config

Package used in getting/setting component properties.

Class Summary

| PropertySetter | General purpose Object property setter. |

org.apache.log4j.config

Class PropertySetter

java.lang.Object

`org.apache.log4j.config.PropertySetter`

public class PropertySetter

extends java.lang.Object

General purpose Object property setter. Clients repeatedly invokes `setProperty(name, value)` in order to invoke setters on the Object specified in the constructor. This class relies on Java reflection.

Usage:

```java
PropertySetter ps = new PropertySetter(anObject);
ps.set("name", "Joe");
ps.set("age", "32");
ps.set("isMale", "true");
```

will cause the invocations `anObject.setName("Joe")`, `anObject.setAge(32)` and `setMale(true)` if such methods exist with those signatures.

Since: 1.1

Constructor Summary

**PropertySetter**(java.lang.Object obj)

Create a new PropertySetter for the specified Object.

Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th>activate()</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>static void</th>
<th>setProperties(java.lang.Object obj, java.util.Properties properties, java.lang.String prefix)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set the properties of an object passed as a parameter in one go.</td>
</tr>
</tbody>
</table>

| void | setProperties(java.util.Properties properties, java.lang.String prefix) |

<table>
<thead>
<tr>
<th>void</th>
<th>setProperty(java.lang.String name, java.lang.String value)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set a property on this PropertySetter's Object.</td>
</tr>
</tbody>
</table>
Constructors

PropertySetter

public PropertySetter(java.lang.Object obj)
Create a new PropertySetter for the specified Object. This is done in preparation for invoking
setProperty(java.lang.String, java.lang.String) one or more times.
Parameters:
obj - the object for which to set properties

Methods

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString,
wait, wait, wait

Constructor Detail

PropertySetter

public PropertySetter(java.lang.Object obj)
Create a new PropertySetter for the specified Object. This is done in preparation for invoking
setProperty(java.lang.String, java.lang.String) one or more times.
Parameters:
obj - the object for which to set properties

Method Detail

setProperty

public void setProperty(java.lang.String name,
java.lang.String value)
Set a property on this PropertySetter's Object. If successful, this method will invoke a setter method on the
underlying Object. The setter is the one for the specified property name and the value is determined partly
from the setter argument type and partly from the value specified in the call to this method.
If the setter expects a String no conversion is necessary. If it expects an int, then an attempt is made to
convert 'value' to an int using new Integer(value). If the setter expects a boolean, the conversion is by new
Boolean(value).
Parameters:
name - name of the property
value - String value of the property

activate

public void activate()
Package org.apache.log4j.spi

Contains part of the System Programming Interface (SPI) needed to extend log4j.

### Interface Summary

| **LoggerRepository** | A LoggerRepository is used to create and retrieve Loggers. |

**org.apache.log4j.spi**

**Interface LoggerRepository**

**All Known Implementing Classes:**

- Hierarchy

```java
public interface LoggerRepository
```

A LoggerRepository is used to create and retrieve Loggers. The relation between loggers in a repository depends on the repository but typically loggers are arranged in a named hierarchy.

In addition to the creational methods, a LoggerRepository can be queried for existing loggers, can act as a point of registry for events related to loggers.

**Since:** 1.2

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addHierarchyEventListener(HierarchyEventListener listener)</td>
<td>Add a HierarchyEventListener event to the repository.</td>
</tr>
<tr>
<td>void emitNoAppenderWarning(Category cat)</td>
<td></td>
</tr>
<tr>
<td>Logger exists(java.lang.String name)</td>
<td></td>
</tr>
<tr>
<td>void fireAddAppenderEvent(Category logger, Appender appender)</td>
<td></td>
</tr>
<tr>
<td>java.utilEnumeration getCurrentCategories()</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>java.utilEnumeration getCurrentLoggers()</td>
<td></td>
</tr>
<tr>
<td>Logger getILogger(java.lang.String name)</td>
<td></td>
</tr>
<tr>
<td>Logger getILogger(java.lang.String name, LoggerFactory factory)</td>
<td></td>
</tr>
<tr>
<td>Logger getRootLogger()</td>
<td></td>
</tr>
</tbody>
</table>
Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level getThreshold()</td>
<td>Get the repository-wide threshold.</td>
</tr>
<tr>
<td>boolean isDisabled(int level)</td>
<td>Returns whether this repository is disabled for a given level.</td>
</tr>
<tr>
<td>void resetConfiguration()</td>
<td></td>
</tr>
<tr>
<td>void setThreshold(Level level)</td>
<td>Set the repository-wide threshold.</td>
</tr>
<tr>
<td>void setThreshold(java.lang.String val)</td>
<td>Another form of setThreshold(Level) accepting a string parameter instead of a Level.</td>
</tr>
<tr>
<td>void shutdown()</td>
<td></td>
</tr>
</tbody>
</table>

Method Detail

addHierarchyEventListener

void addHierarchyEventListener(HierarchyEventListener listener)
Add a HierarchyEventListener event to the repository.

isDisabled

boolean isDisabled(int level)
Returns whether this repository is disabled for a given level. The answer depends on the repository threshold and the level parameter. See also setThreshold(org.apache.log4j.Level) method.

setThreshold

void setThreshold(Level level)
Set the repository-wide threshold. All logging requests below the threshold are immediately dropped. By default, the threshold is set to Level.ALL which has the lowest possible rank.

setThreshold

void setThreshold(java.lang.String val)
Another form of setThreshold(Level) accepting a string parameter instead of a Level.

emitNoAppenderWarning

void emitNoAppenderWarning(Category cat)

getThreshold

Level getThreshold()
Get the repository-wide threshold. See setThreshold(Level) for an explanation.
getLogger
Logger \texttt{getLogger(java.lang.String name)}

getLogger
Logger \texttt{getLogger(java.lang.String name, LoggerFactory factory)}

getRootLogger
Logger \texttt{getRootLogger()}

exists
Logger \texttt{exists(java.lang.String name)}

shutdown
\texttt{void shutdown()}

gGetCurrentLoggers
\texttt{java.util.Enumeration \texttt{getCurrentLoggers()}}

gGetCurrentCategories
\texttt{java.util.Enumeration \texttt{getCurrentCategories()}}
    \texttt{Deprecated. Please use \texttt{getCurrentLoggers()} instead.}

fireAddAppenderEvent
\texttt{void fireAddAppenderEvent(Category logger, Appender appender)}

resetConfiguration
\texttt{void resetConfiguration()}
Package org.apache.log4j.xml

XML based components.

Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOMConfigurator</td>
<td>Use this class to initialize the log4j environment using a DOM tree.</td>
</tr>
</tbody>
</table>

org.apache.log4j.xml

Class DOMConfigurator

java.lang.Object

   org.apache.log4j.xml.DOMConfigurator

All Implemented Interfaces:
   Configurator

public class DOMConfigurator extends java.lang.Object
   implements Configurator

Use this class to initialize the log4j environment using a DOM tree.

The DTD is specified in log4j.dtd.

Sometimes it is useful to see how log4j is reading configuration files. You can enable log4j internal logging by defining the log4j.debug variable on the java command line. Alternatively, set the debug attribute in the log4j:configuration element. As in

   <log4j:configuration debug="true"
   xmlns:log4j="http://jakarta.apache.org/log4j/"
   ...>
   </log4j:configuration>

This version of DOMConfigurator has been modified to support the definition of groups as described in Logger by adding a 'group' element:

   <group name="GROUPNAME" level="LEVEL"/>

All loggers retrieved via Logger.getLogger(String, String) or Logger.getLogger(Class, String) with the same group name will have their effective level set to the provided level.

This group-defined Level verbosity threshold can be overridden by setting an explicit level on the Logger itself.

There are sample XML files included in the package.

Since:
   0.8.3

Field Summary

Fields inherited from interface org.apache.log4j.spi.Configurator

INHERITED, NULL
## Constructor Summary

**DOMConfigurator()**
- No argument constructor.

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td><code>configure(Element element)</code></td>
<td>Configure log4j using a configuration element as defined in the log4j.dtd.</td>
</tr>
<tr>
<td>static void</td>
<td><code>configure(java.lang.String filename)</code></td>
<td>A static version of <code>doConfigure(String, LoggerRepository)</code></td>
</tr>
<tr>
<td>static void</td>
<td><code>configure(java.net.URL url)</code></td>
<td>A static version of <code>doConfigure(URL, LoggerRepository)</code></td>
</tr>
<tr>
<td>static void</td>
<td><code>configureAndWatch(java.lang.String configFilename)</code></td>
<td>Like <code>configureAndWatch(String, long)</code> except that the default delay as defined by <code>FileWatchdog#DEFAULT_DELAY</code> is used.</td>
</tr>
<tr>
<td>static void</td>
<td><code>configureAndWatch(java.lang.String configFilename, long delay)</code></td>
<td>Read the configuration file <code>configFilename</code> if it exists.</td>
</tr>
<tr>
<td>void</td>
<td><code>doConfigure(Element element, LoggerRepository repository)</code></td>
<td>Configure by taking in an DOM element.</td>
</tr>
<tr>
<td>void</td>
<td><code>doConfigure(java.lang.String filename, LoggerRepository repository)</code></td>
<td></td>
</tr>
<tr>
<td>static java.lang.Object</td>
<td><code>parseElement(Element element, java.util.Properties props, java.lang.Class expectedClass)</code></td>
<td>Creates an object and processes any nested param elements but does not call activateOptions.</td>
</tr>
<tr>
<td>static void</td>
<td><code>setParameter(Element elem, PropertySetter propSetter, java.util.Properties props)</code></td>
<td>Sets a parameter based from configuration file content.</td>
</tr>
</tbody>
</table>
Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>

Methods inherited from class java.lang.Object

- clone
- equals
- finalize
- getClass
- hashCode
- notify
- notifyAll
- toString
- wait
- wait
- wait

Constructor Detail

DOMConfigurator

public DOMConfigurator()

No argument constructor.

Method Detail

configure

public static void configure(Element element)

Configure log4j using a configuration element as defined in the log4j.dtd.

Throws:
java.lang.SecurityException - if the caller does not have MonitorAppPermission("logger.config").

configureAndWatch

public static void configureAndWatch(java.lang.String configFilename)

Like configureAndWatch(String, long) except that the default delay as defined by FileWatchdog#DEFAULT_DELAY is used.

Parameters:
configFilename - A log4j configuration file in XML format.

Throws:
java.lang.SecurityException - if the caller does not have MonitorAppPermission("logger.config").

configureAndWatch

public static void configureAndWatch(java.lang.String configFilename, long delay)

Read the configuration file configFilename if it exists. Moreover, a thread will be created that will periodically check if configFilename has been created or modified. The period is determined by the delay argument. If a change or file creation is detected, then configFilename is read to configure log4j.

Parameters:
configFilename - A log4j configuration file in XML format.
delay - The delay in milliseconds to wait between each check.

Throws:
java.lang.SecurityException - if the caller does not have MonitorAppPermission("logger.config").
doConfigure

public void doConfigure(java.lang.String filename,
                        LoggerRepository repository)

    throws java.lang.SecurityException - if the caller does not have
    MonitorAppPermission("logger.config").

doConfigure

public void doConfigure(java.net.URL url,
                        LoggerRepository repository)

    Description copied from interface: Configurator
    Interpret a resource pointed by a URL and set up log4j accordingly. The configuration is done relative to
    the hierarchy parameter.
    Specified by:
    doConfigure in interface Configurator
    Parameters:
    url - The URL to parse
    repository - The hierarchy to operation upon.
    Throws:
    java.lang.SecurityException - if the caller does not have
    MonitorAppPermission("logger.config").

doConfigure

public void doConfigure(java.io.InputStream inputStream,
                        LoggerRepository repository)

    throws javax.xml.parsers.FactoryConfigurationError
    Configure log4j by reading in a log4j.dtd compliant XML configuration file.
    Throws:
    java.lang.SecurityException - if the caller does not have
    MonitorAppPermission("logger.config").
    javax.xml.parsers.FactoryConfigurationError

doConfigure

public void doConfigure(java.io.Reader reader,
                        LoggerRepository repository)

    throws javax.xml.parsers.FactoryConfigurationError
    Configure log4j by reading in a log4j.dtd compliant XML configuration file.
    Throws:
    java.lang.SecurityException - if the caller does not have
    MonitorAppPermission("logger.config").
    javax.xml.parsers.FactoryConfigurationError

doConfigure

public void doConfigure(Element element,
                        LoggerRepository repository)

    Configure by taking in a DOM element.
    Throws:
    java.lang.SecurityException - if the caller does not have
    MonitorAppPermission("logger.config").
configure

public static void configure(java.lang.String filename)
    throws javax.xml.parsers.FactoryConfigurationError

    A static version of doConfigure(String, LoggerRepository).
    Throws:
    java.lang.SecurityException - if the caller does not have
    MonitorAppPermission("logger.config").
    javax.xml.parsers.FactoryConfigurationError

configure

public static void configure(java.net.URL url)
    throws javax.xml.parsers.FactoryConfigurationError

    A static version of doConfigure(URL, LoggerRepository).
    Throws:
    java.lang.SecurityException - if the caller does not have
    MonitorAppPermission("logger.config").
    javax.xml.parsers.FactoryConfigurationError

subst

public static java.lang.String subst(java.lang.String value,
    java.util.Properties props)

    Substitutes property value for any references in expression.
    Parameters:
    value - value from configuration file, may contain literal text, property references or both
    props - properties.
    Returns:
    evaluated expression, may still contain expressions if unable to expand.
    Throws:
    java.lang.SecurityException - if the caller does not have
    MonitorAppPermission("logger.config").
    Since:
    1.2.15

setParameter

public static void setParameter(Element elem,
    PropertySetter propSetter,
    java.util.Properties props)

    Sets a parameter based from configuration file content.
    Parameters:
    elem - param element, may not be null.
    propSetter - property setter, may not be null.
    props - properties
    Throws:
    java.lang.SecurityException - if the caller does not have
    MonitorAppPermission("logger.config").
    Since:
    1.2.15
parseElement

```java
public static java.lang.Object parseElement(Element element,
                                             java.util.Properties props,
                                             java.lang.Class expectedClass)
    throws java.lang.Exception
```

Creates an object and processes any nested param elements but does not call activateOptions. If the class also supports UnrecognizedElementParser, the parseUnrecognizedElement method will be call for any child elements other than param.

**Parameters:**
- element - element, may not be null.
- props - properties
- expectedClass - interface or class expected to be implemented by created class

**Returns:**
- created class or null.

**Throws:**
- java.lang.Exception - thrown if the contain object should be abandoned.
- java.lang.SecurityException - if the caller does not have MonitorAppPermission("logger.config").

**Since:**
1.2.15

doOnChange

```java
public void doOnChange()
```

Call configure(String) with the filename to reconfigure log4j.

**Throws:**
- java.lang.SecurityException - if the caller does not have MonitorAppPermission("logger.config").
Appendix I  Revision History

The following ECN was incorporated into OCAP 1.1.2:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Date Accepted</th>
<th>Title of EC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCAP1.1.1-N-09.1437-1</td>
<td>9/30/09</td>
<td>Make ECN 1035 backward compatible</td>
</tr>
</tbody>
</table>

The following ECNs were incorporated into OCAP 1.1.3:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Date Accepted</th>
<th>Title of EC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCAP1.1.2-N-09.1468-4</td>
<td>6/3/10</td>
<td>Application Download Error Recovery</td>
</tr>
<tr>
<td>OCAP1.1.2-N-09.1485-2</td>
<td>6/3/10</td>
<td>Clarify use of the return value of the notifyXAITUpdate method</td>
</tr>
<tr>
<td>OCAP1.1.2-N-10.1547-1</td>
<td>6/3/10</td>
<td>Java TV 1.1.1 spec reference update</td>
</tr>
<tr>
<td>OCAP1.1.2-N-10.1550-1</td>
<td>6/3/10</td>
<td>TVTimer clarifications</td>
</tr>
</tbody>
</table>

The following changes were incorporated into OCAP 1.1.5:

No ECNs were incorporated into OCAP 1.1.5. The cross-references to other CableLabs specifications were updated and the version number revised to match the OCAP 1.1.5 bundle number. Henceforth, the OCAP version number will always be updated to match the Bundle release version.

The following ECNs were incorporated into OCAP 1.2:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Date Accepted</th>
<th>Title of EC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCAP1.1.5-N-09.1388-6</td>
<td>5/12/11</td>
<td>eCM MIB Query</td>
</tr>
<tr>
<td>OCAP1.1.5-N-10.1554-4</td>
<td>5/12/11</td>
<td>Clarify method queryCapability() in class org.ocap.hardware.VideoOutputPort</td>
</tr>
<tr>
<td>OCAP1.1.5-N-10.1555-3</td>
<td>5/12/11</td>
<td>Clarification of application_mode_descriptor handling in cable mode only hosts</td>
</tr>
<tr>
<td>OCAP1.1.5-N-10.1569-2</td>
<td>5/12/11</td>
<td>Synchronize monApp permissions in OCAP Spec</td>
</tr>
<tr>
<td>OCAP1.1.5-N-10.1573-2</td>
<td>5/12/11</td>
<td>Replace spec STD-096-011 with SCTE 19 and SCTE 53</td>
</tr>
<tr>
<td>OCAP1.1.5-N-10.1579-3</td>
<td>5/12/11</td>
<td>VideoOutputPorts to Physical Connectors Clarification</td>
</tr>
<tr>
<td>OCAP1.1.5-N-10.1593-2</td>
<td>5/12/11</td>
<td>Mandatory support of methods in javax.media.GainControl</td>
</tr>
<tr>
<td>OCAP1.1.5-N-10.1595-2</td>
<td>5/12/11</td>
<td>TVTimer.scheduleTimerSpec javadoc inconsistency</td>
</tr>
<tr>
<td>Identifier</td>
<td>Date Accepted</td>
<td>Title of EC</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>OCAP1.1.5-N-10.1629-2</td>
<td>5/12/11</td>
<td>3DTV Alternative Content Event API</td>
</tr>
<tr>
<td>OCAP1.1.5-N-10.1630-2</td>
<td>5/12/11</td>
<td>3DTV Content Format Listener API</td>
</tr>
<tr>
<td>OCAP1.1.5-N-10.1634-2</td>
<td>5/12/11</td>
<td>Application Authentication Clarification</td>
</tr>
<tr>
<td>OCAP1.1.5-N-11.1643-2</td>
<td>5/12/11</td>
<td>Application Notification of CableCARD Reset</td>
</tr>
<tr>
<td>OCAP1.1.5-N-11.1647-2</td>
<td>5/12/11</td>
<td>XML Subsystem for OCAP: JSR 280</td>
</tr>
<tr>
<td>OCAP1.1.5-N-11.1670-3</td>
<td>5/12/11</td>
<td>OCAP Reference Edits</td>
</tr>
</tbody>
</table>

The following ECNs were incorporated into OCAP 1.2.1:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Date Accepted</th>
<th>Title of EC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCAP1.2-N-11.1678-1</td>
<td>1/12/12</td>
<td>Corrections to S3DConfiguration API</td>
</tr>
<tr>
<td>OCAP1.2-N-11.1691-1</td>
<td>1/12/12</td>
<td>HTTPS support clarification</td>
</tr>
<tr>
<td>OCAP1.2-N-11.1714-3</td>
<td>1/12/12</td>
<td>Logging with log4j</td>
</tr>
<tr>
<td>OCAP1.2-N-11.1721-1</td>
<td>1/12/12</td>
<td>OCAP:SharedResourceUsage changes for NetResourceUsage API</td>
</tr>
<tr>
<td>OCAP1.2-N-11.1723-1</td>
<td>1/12/12</td>
<td>Boot-up Power Mode</td>
</tr>
<tr>
<td>OCAP1.2-N-12.1748-1</td>
<td>1/12/12</td>
<td>S3DFormatTypes Update</td>
</tr>
</tbody>
</table>

The following ECNs were incorporated into OCAP 1.2.2:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Date Accepted</th>
<th>Title of EC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCAP1.2.1-N-12.1758-2</td>
<td>2/24/12</td>
<td>OCAP DTD Update for bundle release 1.2.2</td>
</tr>
<tr>
<td>OCAP1.2.1-N-12.1763-1</td>
<td>2/24/12</td>
<td>Clarification of MIBobject.getData</td>
</tr>
</tbody>
</table>
The following ECN was incorporated into OCAP 1.2.3:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Date Accepted</th>
<th>Title of EC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCAP1.2.1-N-12.1775-1</td>
<td>5/31/12</td>
<td>OCAP DTD Update for bundle release 1.2.3</td>
</tr>
</tbody>
</table>

The following ECNs were incorporated into OCAP 1.3:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Author</th>
<th>Date Accepted</th>
<th>Title of EC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCAP1.2.3-N-11.1652-6</td>
<td>Smith</td>
<td>4/18/2013</td>
<td>Additional OCAP RemoteControl Key Event Codes</td>
</tr>
<tr>
<td>OCAP1.2.3-N-11.1692-4</td>
<td>Ladd</td>
<td>4/18/2013</td>
<td>Get Transport ID</td>
</tr>
<tr>
<td>OCAP1.2.3-N-11.1736-4</td>
<td>Ridley</td>
<td>4/18/2013</td>
<td>Provide access to input video scan mode</td>
</tr>
<tr>
<td>OCAP1.2.3-N-12.1806-1</td>
<td>Ladd</td>
<td>4/18/2013</td>
<td>System Logging</td>
</tr>
<tr>
<td>OCAP1.2.3-N-13.1824-1</td>
<td>Millard</td>
<td>4/18/2013</td>
<td>API Class Clarifications</td>
</tr>
<tr>
<td>OCAP1.2.3-N-13.1839-1</td>
<td>Barringer</td>
<td>4/18/2013</td>
<td>DTD Update for bundle release 1.3</td>
</tr>
</tbody>
</table>

The following ECNs were incorporated into OCAP 1.3.1:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Author</th>
<th>Date Accepted</th>
<th>Title of EC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCAP1.3-N-12.1799-2</td>
<td>Dulapa</td>
<td>5/30/13</td>
<td>Change DVBGraphics Extension</td>
</tr>
<tr>
<td>OCAP1.3-N-13.1844-1</td>
<td>Millard</td>
<td>5/30/13</td>
<td>Throw IllegalArgumentException on Invalid Locator</td>
</tr>
<tr>
<td>OCAP1.3-N-13.1849-1</td>
<td>Barringer</td>
<td>5/30/13</td>
<td>Permission Request File DTD for Bundle 1.3.1</td>
</tr>
</tbody>
</table>

End of OCAP