Release Notes for OCAP1-Full Source Version 1.0-ER2

Version 1.0-ER2
March 29, 2007

Release Notes for OCAP1-Full Source Version 1.0-ER2
Publication Date: March 29, 2007

Copyright and reproduction notice

Confidentiality and proprietary notice

Confidential and proprietary information subject to non-disclosure agreements with OCAP Development LLC. This information is to be used solely for evaluation of continued and future business agreements with OCAP Development LLC, and may not be distributed or copied without written permission from OCAP Development LLC.

Warranty disclaimer

This publication is provided "As Is" without any warranty of any kind, either express or implied. This includes, but is not limited to, the implied warranty of merchantability, fitness for a particular purpose, or non-infringement.

This publication could include technical inaccuracies or typographical errors. Changes made to this software between documentation releases will be incorporated into new editions of the publication. OCAP Development LLC and/or Vidiom Systems, Inc. may make improvements and/or changes in the products described in this publication at any time.

Contents

1 Overview

2 v1.0-ER2 Release Information

3 v1.0-ER1 Release Information

Preface

Overview

These release notes are for the v1.0-ER2 release of the OCAP1 software. This section provides:

- *Conventions* used in this document

Conventions

This manual uses the following icons to alert you to special information:

NOTE: This is a note. Notes contain important information that is not procedurally dangerous.

CAUTION: This is a documentation alert. It contains information about unusual behavior of a utility, command, or procedure.

Cross references provide information about where you can find related information within the manuals provided for the VISION Workbench. The cross references are located on the right side of the text.

This is a cross reference to related information.

Definitions of words or phrases that are used in the text are also located on the right side of the text.

This is a definition of a word or phrase found in the text that you may not be familiar with, or that may be used differently for VISION Workbench.

In addition, the following typographic conventions and symbols are used in this manual:

- The names of commands, files, and directories, as well as on-screen computer output, are indicated by using the following font:

  `AaBbCc123`

The following are examples of this typographical convention:

> This definition is found in the `psm.h` file.
>
> Use `ls -a` to list all files.

- The following font indicates that you can replace the name of a variable with an actual name or value:

  AaBbCc123

  The following is an example of this typographical convention:

  > Use the *mpegid* parameter to identify the specific MPEG decoder.

- Optional elements are contained within square brackets ([]). For example:

  `sometool [`*optional*`]`

- Groupings of elements are enclosed in braces ({}). For example:

  ```
  { CENTER | TOP | BOTTOM }
  ```

- To repeat the previous item in a syntax statement, ellipses (. . .) are used. For example:

  `sometool [`*option*` ...]`

- A vertical bar (|) is used both for the logical OR in syntax statements and to indicate a choice of two items. For example:

  ```
  { CENTER | TOP | BOTTOM }
  ```

1 Overview

Overview

This chapter contains the following information:

- *Documentation overview*
- *Contents of package*
- *Technical support*

Documentation overview

The release notes cover new features, as well as limitations. The following are brief descriptions of all the chapters included in the release notes:

- Chapter 1, *Overview*, provides contact information for Vidiom technical support.
- Chapter 2, *Release Information*, provides changes to the OCAP1 release between v1.0-ER1 and v1.0-ER2.
- Appendix A, *v1.0-ER1 Release Information*, provides changes to the OCAP1 release between v0.9.5 and v1.0-ER1.

Contents of package

The Full Source release CD includes the following components:

- Full source of the OCAP Java libraries
- Full source and binaries of the Java integration tests
- Full source and binaries of the JUnit tests
- OCAP Porting Kit
- Release notes
- Build tools necessary to compile the OCAP Java libraries

✦ **NOTE:** Vidiom does not provide ALL the build tools. Some are only referenced because we cannot OEM them. The OCAP Porting Guide contains a complete list of tools.

Technical support

For help with the OCAP1 version 1.0-ER2, you may contact technical support at Vidiom by sending an email to:

```
support@vidiom.com
```

Please include a description of the problem that you are experiencing, along with the part number on your CD-ROM and the version number of your software.

The part number for version 1.0-ER2 of the Full Source release is as follows:

```
VS-ODL-OCAP1-FS-1.0 ER-2
```

2 v1.0-ER2 Release Information

Overview

This section describes the changes that were made between the v1.0-ER1 and v1.0-ER2 releases. It covers the following topics:

- *Build changes*
- *Known bugs*
- *Supported platforms and dependencies*
- *OCAP I16 compliance*
- *Currently supported OCAP APIs*
- *Closed captioning: VBI filtering*
- *DVR: Media time tags*
- *DVR: Storage management*
- *Graphics resolutions*
- *JMF improvements*
- *Media improvements*
- *MediaStorageVolume.allowAccess()/removedAccess()*
- *mpeenv.ini configuration*
- *Persistent memory*
- *Stream events*
- *Video-drip support*

Build changes

This release is being provided as an integral part of the Scientific Atlanta Axiom code release. It is not being distributed as a separately installable package at this time.

Known bugs

This section provides information about known P1 and P2 bugs.

Digital Video Recorder (DVR)

The following are the known DVR bugs:

ID	Summary
2637	`ExtendedResourceUsages`, used for scheduled recordings, references the caller context of the application which scheduled the recording. However, the scheduling application may not be around when the recording begins (because of rebooting, the application has been destroyed, etc.). This causes inconsistencies with resource management at the time the recording begins, or when other recordings are scheduled
5323	OCAP stack seems to lock up on the playback of a Recorded Service that had been terminated with a tune away
5364	Frame stepping in DVR does not appear to be functional
5368	`ClassCastException` thrown on creating parent root

File system

The following is the known file system bug:

ID	Summary
5096	`LoadedFileSys` returns potentially wrong values

Media timer

The following is the known media timer bug:

ID	Summary
5361	`getMediaTime` fails to adjust after `setMediaTime` is called while at a rate of `0`

Object carousel

The following are the known object carousel bugs:

ID	Summary
3254	There is a possibility the set-top box will be unable to read some sections in large modules on in-band object carousels
3630	When the carousel operates in a non-monitoring mode with regard to the Dynamic Invocation Interface (DII), it holds a global lock. Although we are unlikely to see a problem, there is a possibility a problem would occur if the carousel goes away, and attempts to read the carousel would cause all carousel activity to stall out for the time-out period

PAT/PMT

The following are the known Program Association Table (PAT)/Program Map Table (PMT) bugs:

ID	Summary
4266	OCAP stack does not provide time-shifted service information/PMT
4561	Out-Of-Band (OOB) PAT/PMT `TableChangeListeners` sometimes fails to pick up Packet Identification (PID) changes
5029	`ServiceContextImpl` fails to send out `NormalContentEvent` after successfully tuning to PAT/PMT transmission
5394	Primary audio comes back corrupted on a PMT change when the background and PIP window tuned to the same service
5398	PMT audio PID change occasionally not being reflected when the Primary and PIP window are tune to the same service

PowerTV

The following are the known PowerTV bugs:

ID	Summary
3525	Early halt of playback before the end of recording
3820	Capture manager uses incorrect TVPID. When a new tune is done with the second tuner, the player or video stream associated with the `BackgroundVideoPresentationControl` gets switched to the new player/stream
3935	Assert failed in `avfs_SelectPartitionForRec` on tune when the disk is full. PowerTV does not stop ongoing recordings with enough space to keep the set-top box from crashing
3952	Problem tuning to digital channels. PowerTV could not find Packet Identification (PID) information for the service
4710	Closed Captioning does not work
5074	Crashes when changing from standard-definition to high-definition programming

ID	Summary
5109	PowerTV crash while running TuneTest
5139	PowerTV operating system (TVP manager) not receiving `FirstFrame` event
5347	`RandomAccessFile` created in persistent storage does not allow read after extending length by write

Section filtering

The following is the known section filtering bug:

ID	Summary
3091	Section filtering pending read list `sectionsRemaining` count is off

Switched digital video

The following is the known switched digital video bug:

ID	Summary
5080	`SwitchedDigitalVideo` - new locator does not automatically reselected (no `PresentationChangedEvent`) with `modifyService()`

Time-shift buffer (TSB)

The following are the known TSB bugs:

ID	Summary
4233	Standard definition/high-definition switching appears to shutdown the TSB
5445	Assertion failure exception on constructor of `BroadcastSession`

Tuning

The following is the known tuning bug:

ID	Summary
5383	`OutOfMemoryException` occurring on buffered tuning

Supported platforms and dependencies

SA-8300

The SA-8300HD port now uses the PowerTV operating system version ADK61511_2p-Explorer8kg3-ATSC-SA-pKey.

OCAP I16 compliance

The unimplemented OCAP Engineering Change Numbers (ECN) and Engineering Change Orders (ECO) (limitations from the OCAP I16 specification (OC-SP-OCAP1.0-I16)) are as follows:

ECN/ ECO	Description
645	Provide a Vertical Blanking Interval (VBI) data retrieval function (I12)
661	Clarify a rule of video/graphics resolution
911	Media time tags

Currently supported OCAP APIs

This section provides information about the currently supported OCAP Application Programming Interfaces (APIs).

Core Java APIs (J2ME/pJava)

The following list provides the core Java API status:

Core Java APIs (J2ME/pJava)	Package Implemented?	Notes
`java.awt`	yes	
`java.awt.event`	yes	
`java.awt.image`	yes	
`java.beans`	yes	
`java.io`	yes	
`java.lang`	yes	
`java.lang.reflect`	yes	
`java.math`	yes	
`java.net`	yes	
`java.rmi`	yes	
`java.security`	yes	
`java.security.cert`	yes	
`java.security.spec`	yes	
`java.util`	yes	
`java.util.zip`	yes	

JMF-1.0

The following list provides the Java Media Framework (JMF)-1.0 API status:

JMF-1.0 APIs	Package Implemented?	Notes
`javax.media`	yes	
`javax.media.protocol`	yes	

JSSE-1.02

The following list provides the Java Secure Socket Extension (JSSE)-1.02 API status:

JSSE-1.02 APIs	Package Implemented?	Notes
`javax.net`	yes	
`javax.net.ssl`	yes	
`javax.security.cert`	yes	

JavaTV 1.0

The following list provides the core JavaTV 1.0 API status:

JavaTV APIs	Package Implemented?	Notes
`javax.tv.graphics`	yes	
`javax.tv.locator`	yes	
`javax.tv.media`	yes	
`javax.tv.net`	yes	
`javax.tv.service`	yes	Exceptions: `SIManager.registerInterest()` can be called, but has no effect. Program Event support not implemented because it is defined by Society of Cable Telecommunications Engineers (SCTE) profiles 4 and 5 which are not currently used by any head-end
`javax.tv.service.guide`	yes	Exceptions: Currently support profiles 1 and 2. Support for profiles 3, 4, 5 not implemented because no head-ends in U.S. currently provides Event Information Table (EIT) service information data

JavaTV APIs	Package Implemented?	Notes
javax.tv.service.navigation	yes	As per OCAP Specification, CAIdentification should not be used by applications
javax.tv.service.selection	yes	
javax.tv.service.transport	yes	
javax.tv.util	yes	
javax.tv.xlet	yes	

DAVIC-1.4.1p9

The following list provides the Digital Audio Visual Council (DAVIC)-1.4.1p9 API status:

DAVIC-1.4.1p9 APIs	Package Implemented?	Notes
org.davic.media	yes	
org.davic.mpeg	yes	
org.davic.mpeg.sections	yes	
org.davic.net	yes	
org.davic.net.tuning	yes	
org.davic.resources	yes	

MHP-1.0.3

The following list provides the Multimedia Home Platform (MHP)-1.0.3 API status:

MHP-1.0.3 APIs	Package Implemented?	Notes
org.dvb.application	yes	
org.dvb.dsmcc	partial	Mostly complete. Support for DSMCC stream events is incomplete (especially scheduled events).
org.dvb.event	yes	
org.dvb.io.ixc	yes	
org.dvb.io.persistent	yes	
org.dvb.lang	yes	
org.dvb.media	yes	
org.dvb.net	yes	
org.dvb.net.rc	yes	

MHP-1.0.3 APIs	Package Implemented?	Notes
`org.dvb.net.tuning`	yes	
`org.dvb.test`	yes	All `DVBTest` methods throw `IOException` exceptions as this class is not currently being used for testing an OCAP stack (`org.ocap.test` is used instead)
`org.dvb.ui`	yes	
`org.dvb.user`	yes	

HAVi-1.1

The following list provides the Home Audio Video Interoperability (HAVi)-1.1 API status:

HAVi-1.1 APIs	Package Implemented?	Notes
`org.havi.ui`	yes	
`org.havi.ui.event`	yes	Virtual keyboard support is not required and not implemented.

OCAP-1.0 (I16)

The following list provides the OCAP-1.0 API status:

OCAP-1.0 APIs	Package Implemented?	Notes
`org.ocap`	yes	
`org.ocap.application`	yes	
`org.ocap.event`	yes	
`org.ocap.hardware`	yes	
`org.ocap.hardware.pod`	yes	Stack support is complete. Port must be customized to specific platform
`org.ocap.media`	partial	Vertical Blanking Interval (VBI) data retrieval is not implemented
`org.ocap.mpeg`	yes	
`org.ocap.net`	yes	
`org.ocap.resource`	yes	
`org.ocap.service`	yes	
`org.ocap.si`	yes	
`org.ocap.storage`	yes	
`org.ocap.system`	yes	

OCAP-1.0 APIs	Package Implemented?	Notes
`org.ocap.system.event`	yes	
`org.ocap.test`	yes	
`org.ocap.ui.event`	yes	

OCAP DVR I02

The following list provides the OCAP DVR I02 API status:

OCAP DVR APIs	Package Implemented?	Notes
`org.ocap.dvr`	partial	Exception: Complete except recorded applications will be post v1.0
`org.ocap.shared.dvr`	partial	Exception: Complete except DVR time-line support will be added by Scientific Atlanta post this release
`org.ocap.shared.dvr.navigation`	yes	
`org.ocap.dvr.storage`	yes	
`org.ocap.shared.media`	partial	Exception: Complete except DVR time-line support will be implemented by Scientific Atlanta post this release

OCAP front panel extension

The following list provides the OCAP front panel extension API status:

OCAP front panel API	Package Implemented?	Notes
`org.ocap.hardware.frontpanel`	yes	

OCAP home networking extension

The following list provides the OCAP home networking extension API status:

OCAP home networking API	Package Implemented?	Notes
`org.ocap.hn`	no	Post v1.0
`org.ocap.hn.content`	no	Post v1.0
`org.ocap.hn.content.navigation`	no	Post v1.0
`org.ocap.hn.services`	no	Post v1.0
`org.ocap.hn.util`	no	Post v1.0

OCAP ScaledVideoMgr extension

The following list provides the OCAP `ScaledVideoMgr` extension API status:

OCAP scaled video API	Package Implemented?	Notes
`org.ocap.system.ScaledVideoManager`	partial	Parts of the draft ECR for this package were implemented. This package will likely be deprecated in deference to the new `MultipleScreenManager` facility being added to the core OCAP 1.1 specification (still in an ECR state)

Closed captioning: VBI filtering

OCAP VBI data retrieval package is defined by OCAP ECN645. Implementation of this package is incomplete, but public APIs are currently stubbed out.

This release introduces a session-centric mechanism to identify decoding activities and corresponding resources.

`mpeos_mediaPause()` was removed.

Currently, `mpe_mediaDecode()` and `mpe_mediaStop()` were adapted. Now function `mpeos_mediaDecode()` returns a session parameter and `mpeos_mediaStop()` takes a session instead of a device identifier parameter. For detailed descriptions, refer to the OCAP Porting Guide. The new functions are defined as follows:

```
mpe_Error mpeos_mediaDecode(
   mpe_MediaDecodeRequestParams *decodeRequest,
   mpe_EventQueue queueId,
   void *act,
   mpe_MediaDecodeSession *session )
mpe_Error mpeos_mediaStop(
   mpe_MediaDecodeSession session );
```

DVR: Media time tags

The new media time tag functionality (ECN911) allows OCAP applications to create and listen for trigger events based on proprietary trigger information. The information is found by the application in the live broadcast. When the corresponding play position is crossed during normal playback and trick mode presentation, content tuned by the set-top box is stored and deliver by the OCAP middleware to interested applications.

The media time tag changes affected the following areas of the OCAP stack:

- OCAP packages, classes, and interfaces
- Axiom-base Java modules
- Axiom DVR Java modules

OCAP packages, classes, and interfaces

The following OCAP packages, classes, and interfaces were added or modified:

```
org.ocap.dvr.event.LightweightTriggerManager (new class)
org.ocap.dvr.event.LightweightTriggerHandler (new interface)
org.ocap.dvr.event.LightweightTriggerSession (new interface)
org.ocap.dvr.event.StreamChangedListener (new interface)
org.ocap.dvr.OcapRecordingManager (added methods)
org.ocap.dvr.RecordingPlaybackListener (new interface)
org.dvb.dsmcc.ServiceDomain
   (enhanced functionality, no interface changes)
org.dvb.dsmcc.DSMCCObject
   (enhanced functionality, no interface changes)
org.dvb.dsmcc.DSMCCStream
   (enhanced functionality, no interface changes)
org.dvb.dsmcc.DSMCCStreamEvent
   (enhanced functionality, no interface changes)
```

Axiom-base Java modules

The following Axiom-base Java modules were affected:

```
org.dvb.dsmcc.ServiceDomain
org.dvb.dsmcc.DSMCCObject
org.dvb.dsmcc.DSMCCStream
org.dvb.dsmcc.DSMCCStreamEvent
com.vidiom.impl.manager.MediaTimeTagsManager (new interface)
com.vidiom.impl.manager.mtt.MockMediaTimeTagsMgrImpl (new)
com.vidiom.impl.manager.mtt.MediaTimeTagsChangeListener
   (new interface)
```

Axiom DVR Java modules

The following Axiom DVR Java modules were affected:

```
com.vidiom.impl.manager.recording.RecordingManagerImpl
com.vidiom.impl.manager.mtt.LightweightTriggerManagerImpl
   (new)
com.vidiom.impl.manager.mtt.LightweightTriggerSessionExt
   (new interface)
com.vidiom.impl.manager.mtt.LightweightTriggerManagerExt
   (new interface)
com.vidiom.impl.manager.mtt.LightweightTriggerSessionImpl
   (new)
com.vidiom.impl.manager.mtt.MediaTimeTagsManagerExt
   (new interface)
com.vidiom.impl.manager.mtt.MediaTimeTagsMgrImpl (new)
com.vidiom.impl.manager.recording.RecordedServiceImpl
com.vidiom.impl.recording.RecordingInfo
com.vidiom.impl.manager.TimeShiftManager
com.vidiom.impl.manager.timeshift.TimeShiftManagerImpl
com.vidiom.impl.manager.timeshiftTimeShiftNIProperties
   (new class)
com.vidiom.impl.manager.mtt.MediaTmeTagEvent (new class)
```

DVR: Storage management

The new storage management includes the following types of storage functionality:

Logical storage volumes
Media storage volumes
HDD repartitioning/ reformatting
Detachable storage devices
Free-space listeners

Logical storage volumes

Logical storage volumes were added to allow applications to create their own exclusive directory on a specified storage device and can be used for general-purpose file storage.

NOTE: Logical storage volumes on the SA-8300HD and SA-8300HDC platforms are only a single storage device presented up to the Java middleware layer, which is presented to applications as a single `StorageProxy`.

Detachable storage devices

Detachable storage device functionality was added to deliver an event to interested applications whenever the storage device is connected or disconnected. The event indicates a status change has occurred affecting the storage proxy for the internal Hard-Disk Drive (HDD). From the application's perspective, the only observable change will be a change in capacity and free space.

Media storage volumes

Media Storage Volumes (MSVs) were added to allow applications running on a DVR-capable, set-top box to create their own specialized logical storage volume on a specified storage device. Where the storage device controls all access rights to for storage of media files recorded via the OCAP DVR APIs. The application is also allowed to reserve disk space for their MSVs and the reservation is enforced. When issuing a recording request, applications specify the destination MSV to which the recording should be stored. If no MSV is specified, the default MSV is used.

◆ **NOTE:** MSVs are logical entities rather than physical entities. The Java layer is unaware and this implementation is localized to the MPEOS layer and therefore better portability of the OCAP middleware is retained.

When setting a reservation for an MSV, the minimum size allowed is based on the following calculation (on the SA-8300HD and SA-8300HDC platforms this calculates to ~14MB):

```
ThresholdSizeKB = POLLING_INTERVAL_SEC * MAX_TUNERS
   * (MAX_BIT_RATE_KBITS_PER_SEC / 8)
```

If an application tries to set an MSV reservation for less than this value, it is adjusted to this value.

Free-space listeners

Free-space listeners were added to allow applications to register for notification when the free space on a specific MSV drops below a specified threshold level.

HDD repartitioning/ reformatting

HDD repartitioning/reformatting was added to allow highly privileged applications to control how much space on a specified storage device should be reserved for storing DVR recorded media versus general purpose file storage by repartitioning of the device. Highly privileged application can also explicitly reformat the HDD.

◆ **NOTE:** There is only one storage device and `StorageProxy` presented up to the application, therefore, when an application chooses to re-partition or re-format this `StorageProxy`, it affects both the internal and external HDD. Repartitioning the `StorageProxy` will result in content loss on both the internal and external drives. The ITFS partition on the SA-8300HD or SA-8300HDC platform is limited to 1-4 GB. The MEDIAFS partition size may be adjusted to meet this restriction.

APIs were added to allow applications to query both the overall capacity and free space of a `StorageProxy`, as well as the capacity and free space for media storage versus general purpose data storage.

File access and extended file access permissions were added for logical storage volumes and MSVs.

Functionality was added to the OCAP stack so applications can manage files and recordings in their own space on the HDD.

Graphics resolutions

Portable Java portions of the OCAP stack were updated to support dynamic, graphics device resolution changes based upon configuration changes (ECN661). Native support (in the form of the MPEOS display manager implementation) is required for dynamic, graphics device resolution change. This change simply ensures that the full resolution is visible (for example, not accidentally clipped) and available to applications.

Currently, the PowerTV-based and DirectFB-based MPEOS implementation does not support dynamic, graphics device resolution changes.

Update video configurations

Since I16 the 480i and 480p (4x3) video configurations have changed. MPEOS has been updated to comply with this change and report a pixel resolution of 720x480 instead of 640x480 for both configurations. The resolution of the 480p, 16x9 configuration was also changed from 853x480 to 720x480. The new aspect ratio is as follows:

Video configuration	Screen aspect ratio	Device resolution	Pixel aspect ratio
480i	4:3	720x480	8:9
480p, 4x3	4:3	720x480	8:9
480p, 16x9	16:9	720x480	32:27

◆ **NOTE**: The pixel aspect ratio had to be adjusted in order to obtain the desired video resolution.

JMF improvements

Considerable improvement was made in JMF reliability and robustness. Numerous JMF-related bugs were fixed. The following list is a summary of bug fixes grouped by functional area:

Audio

The following audio bug has been addressed:

4357	`SoundMgrImplTest.testPlaySoundException` fails

Closed captioning

The following closed captioning bugs were addressed:

4728	`ClosedCaptioningControl` not OCAP I16 compliant
5258	`IllegalArgumentException` setting the closed captioning service number in `ClosedCaptioningControl`
5259	Duplicate registrations of a `ClosedCaptioningListener` should be ignored

DVR and trick-mode playback

The following DVR and trick-mode playback bugs were addressed:

4605	`TSBDataSource` returning incorrect time values
4616	`TSBPlayer.getDuration` throws an `NullPointerException` when the player is closed
4752	`AbstractPlayer.setStopTime` doesn't activate alarm
4754	`AbstractPlayer.setStopTime` does not create an alarm when playing backwards
4765	`ServiceContext` fails to present buffered `ServiceContext` in `TestBufferedSC_w_RecByLoc_A - setMediatTime` back 30 seconds
4779	Analog service selection fails frequently with iTSB enabled
4782	Failure to throw a `PresentationTerminatedEvent` when implicitly deleting a `SegmentedRecordedService` while in playback
4875	Presentation terminates when pausing TSB
5138	Rate change not preserved when jumping between segments in a recording playback
5123	Clock time continues to increase when `player.getMediaTime()` is on a playback of recorded service

General

The following general bugs were addressed:

4760	`DVBMediaSelectControl.select` throws index out of bounds on an empty array
4854	Static reference to `SIManager` in `ServiceAdapter` causes test failures
5127	Eliminate remaining vestiges of JMF/JavaTV reference implementations

Network

The following network bug has been addressed:

3061	`NetworkInterface.tune()` should always call native tune

Player resources

The following player resource bugs were addressed:

4780	`NullPointerException` in `AbstractPlayer` when `ServiceContext` is destroyed
5105	JMF player should not reserve `HVideoDevice` with null `CallerContext`
5247	Problem finding available NI when starting application player
5256	Unexpected `ServiceRemovedEvent` with application created player

Service bugs

The following service and PMT bugs were addressed:

5035 Failure to stop presenting a stream when PMT PID for service is removed from PAT

5042 Service re-select occurring on every PMT change regardless of what changes have occurred

5044 PIP window not properly updating to PAT/PMT changes - component tag and stream type changes

5110 JMF not switching to live when handling a PAT/PMT change

5177 Pausing after PMT change fails to present buffered video content

5252 `ServiceContext.select(Locator[])` should accept a service locator

Video presentation

The following video presentation bugs were addressed:

4703 Clipping handled incorrectly in OCAP stack

4773 `NullPointerException` in `HDVideoDevice` when attempting to swap with the `ScaledVideoManager`

4689 `AbstractVideoPlayer` not saving size settings

5095 DFC `ToScalingBounds()` returns incorrect value for `DFC_PROCESSING_LB_14_9`

5146 DFC not applied when starting/restarting the player

5155 Set `ScaledVideoBounds` is using incorrect default clipping region

5066 `VideoPresentationControl.get*Area*()` methods incorrect

5101 `getBoxSizes_*` methods in `AbstractVideoPlayer` need update

5160 `VideoFormatControl.isPlatform()` implementation incorrect

Media improvements

The following bug fixes and improvements were made in the `mpe_media` layer:

4832 `mpe_mediaSetBounds` does not clip/scale correctly when destination is full screen

4833 `mpe_mediaSetBounds` applies incorrect scaling factor for high definition

4890 DFC and Active Format Descriptor (AFD) events not signaled by MPE

5068	TestRecordingPlaybackUsingJMF passes bug fails to present video (video device invalid or inactive)
5106	mpe_dispSetDFC does not support DFC_PLATFORM
5174	Set bounds on analog service before decoding fails
5333	Media decode sessions are not freed after tune failure

Media API refactoring

Portions of the Media API were updated to enable support for new features such as drip feed and VBI (Analog Closed Captioning), as well as to enable a more robust API and implementation. The update includes the introduction of a media decode session that is used by both broadcast decoding (mpe_mediaDecode) and by drip feed (mpe_mediaDripFeedStart, mpe_mediaDripFeedRenderFrame, mpe_mediaDripFeedStop).

The drip-feed API changes are listed in the *Video-drip support* section.

The media decode session updates are as follows:

A media decode session type definition was added and used by new drip-feed methods and is now used in broadcast decode methods (mpe_mediaDecode, mpe_mediaStop). mpe_MediaDecodeSessionH is defined as follows:

```
/* media decode session handle */
typedef struct _mpe_MediaDecodeSessionH { int unused1; }
    *mpe_MediaDecodeSession;
```

The mpe_mediaDecode method was updated to add an output parameter (pointer to a media decode session) to be filled in with a decode session handle. mpe_mediaDecode is defined as follows:

```
/**
* The mpeos_mediaDecode() function shall start
* presenting the media given the pids
*
* @param decodeRequest decode request parameters including pids
*   (audio, video) to select and the mpe_DispDevice param that
*   will present the selected media. Also includes the tuner Id
*   to bind the display device to.
* @param queueId to post decode related events
* @param act is a context value for the event dispatcher
* @param session is a pointer to a media decode session that is
*   filled in by this function
*
* This is an asynchronous operation and returns MPE_SUCCESS if
* successful otherwise appropriate error code is returned. If
* an error is returned no future events will be delivered.
*/
mpe_Error mpeos_mediaDecode (
    mpe_MediaDecodeRequestParams *decodeRequest,
    mpe_EventQueue queueId,
    void *act,
    mpe_MediaDecodeSession *session);
```

The mpe_mediaStop function was updated to change the mpe_DispDevice parameter to mpe_MediaDecodeSession. mpeos_mediaStop is defined as follows:

```
/**
* The mpeos_mediaStop() function shall stop presenting
* the media
*
* @param session media decode session for which the media
*   presentation needs to be stopped
*
* Returns MPE_SUCCESS if successful otherwise appropriate error
* code is returned.
*/
mpe_Error mpeos_mediaStop (mpe_MediaDecodeSession session);
```

The mpe_mediaPause routine has been removed as there was no implementation and it was not exposed to Java. It was an old API that is no longer valid.

MediaStorageVolume.allowAccess()/removedAccess()

Additions to the MediaStorageVolume were made in this release to complete the implementation of ECN929-3. Specifically, the methods allowAccess() and removeAccess() were completed to notify mediaStorageVolume accessors of its disabling or re-enabling. Changes to the disable notification implementation in recording mode was also added to query the MediaStorageVolume for recording owner organization accessibility. Disabling also prevents java.io access to the MediaStorageVolume.

mpeenv.ini configuration

For detailed information regarding environment variables in the mpeenv.ini file, refer to Configuring the mpeenv.ini file section in the OCAP Porting Guide. The following are new environment variables for this release:

OCAP.http.filename.hasdot=*switch*

determines whether to treat the "." as part of a file name. Directory names do not contain "." characters. Where:

switch specifies whether the "." is supported. Possible values for *switch* are:

TRUE enables "." support. TRUE is the default.

FALSE disables "." support.

For example:

```
OCAP.http.filename.hasdot=true
```

`OCAP.appstorage.dvb.use=`*switch*

determines whether to read `dvb.hashfile` if `ocap.hashfile` is not found. Where:

switch specifies whether to read `dvb.hashfile`. Possible values for *switch* are:

`TRUE` reads `dvb.hashfile`.

`FALSE` does not read `dvb.hashfile`. `FALSE` is the default.

For example:

```
OCAP.appstorage.dvb.use=false
```

`OCAP.appstorage.dotdir.use=`*switch*

determines whether to use a `.dir` file or a `.hashfile` to detect a directory. Where:

switch specifies whether to read `dvb.hashfile`. Possible values for *switch* are:

`TRUE` uses a `.dir` file to detect a directory.

`FALSE` uses a `.hashfile` to detect a directory.

For example:

```
OCAP.appstorage.dotdir.use=false
```

`OCAP.mgrmgr.MediaTimeTags=`*manager*

specifies the media, time-tags manager. Where:

manager

identifies the manger.

For example:

```
OCAP.mgrmgr.MediaTimeTags=com.vidiom.impl.manager
   .mtt.MediaTimeTagsMgrImpl
```

Persistent memory

The persistent memory rules as per ECN944 are implemented in this release. ECN944 modifies the rules by which a file can be removed from persistent storage so application priority is considered before file priority. ECN944 also adds APIs that allow an application to query the total amount of persistent storage and the amount of free persistent storage. An application can also add listeners to be notified for a high water mark reached in available persistent storage.

Stream events

Stream event (`DSMCCStreamEvent.subscribe()` and the associated callbacks) functionality has been partially implemented. This is a mechanism for coarse grained, real-time signaling/synchronization of media and applications. An application can open a stream event object out of the object carousel, subscribe to the individual events, and receive a callback when those events occur.

The following limitations apply:

- Only do-it-now events are currently supported. Scheduled events were implemented, but not tested and should not be used.
- Filtering for stream events does not survive tune away and tune back. If you tune away and back, the user will need unsubscribe and re-subscribe.
- `SimpleSectionFilter` receives stray event, stops filtering, and prevents multiple stream events from being filtered correctly. After a few stream events were received, the `DSMCCStreamEvent` object will no longer receive events, and stop signaling them.
- Users should be sure to unsubscribe all events upon Xlet shutdown, as shutdown semantics have not yet been implemented.
- Testing is not complete.

NOTE: An MPE and MPEOS interface still needs to be implemented for stream events requiring normal play time.

Video-drip support

An MPEG-2 video-drip feed is a series of MPEG 2 video frames decoded and displayed with a minimum time delay (>500 ms) between each frame. Data flow is driven by the application using a `DripFeedDataSource` associated with a media player. The application must retrieve the video frames and submit them to the `DripFeedDataSource` one at a time as an array of bytes. Drip-feed applications must be signed and use a permission request file to indicate permission to access a drip feed.

To support video-drip feeds, both the native and Java layers were updated.

Porting layer (native) API additions

The following native methods and data structures were added to the porting layer to support video drip feeds.

◆ **Note:** Drip feed uses a media decode session which is described in detail in the *Media API refactoring* section.

```
/* media decode session handle */
typedef struct _mpe_MediaDecodeSessionH { int unused1; }
*mpe_MediaDecodeSession;

/*
* Parameters for starting a drip feed. This structure is in
* place to serve as a placeholder for future parameters needed
* to start a drip feed. Currently, the only parameter needed is
* a video device for the drip feed output. Future parameters
* may include the clipping and destination regions for scaling.
*/
typedef struct _mpe_MediaDripFeedRequestParams {
mpe_DispDevice videoDevice;
} mpe_MediaDripFeedRequestParams;

/**
* Initialize a drip feed decode session based on the input
* parameters. The decode session can be used to later submit
* frames of data for decode and to stop the decode. For now,
* other aspects of the drip feed (such as bounds) can be
* manipulated using the associated video device with other
* mpe_media routines. In the future, the intent is to have all
* media routines accept a decode session.
*
* @param dripFeedRequest the drip feed request parameters for
* starting the drip feed
* @param eventQueue the event queue to post drip feed events.
* @param act (async completion token) the completion token for
* async events
* @param mediaDecodeSession the session filled in by the method
* to id the drip feed
* @return
* MPE_EINVAL - an input parameter is invalid (null pointer)
* MPE_ERROR_MEDIA_RESOURCE_BUSY - All decode sessions are in
* use
* MPE_SUCCESS - the function completed successfully
*/
mpe_Error mpeos_mediaDripFeedStart(
   mpe_MediaDripFeedRequestParams *dripFeedRequest,
   mpe_EventQueue eventQueue,
   void *act,
mpe_MediaDecodeSession *mediaDecodeSession);
```

```
/**
* Render a single MPEG-2 Video frame synchronously. The expected
* is use is to receive I or P-frames that do not require future
* frames for decode.
* @param mediaDecodeSession the drip feed session created by
* mpe_mediaDripFeedStart
* @param buffer the byte array the contains the MPEG-2 I-Frame
* or P-Frame
* @param length the number of bytes in the byte array
* @return
* MPE_EINVAL - an input parameter is invalid (null pointer)
* MPE_SUCCESS - the function completed successfully
*/
mpe_Error mpeos_mediaDripFeedRenderFrame(
   mpe_MediaDecodeSession mediaDecodeSession,
   uint8 *buffer,
   size_t length);

/**
* Stop a drip feed decode session that was started with
* mpe_mediaDripFeedStart
* @param mediaDecodeSession the session filled in by the method
* to id the drip feed
* @return
* MPE_EINVAL - an input parameter is invalid (null pointer)
* MPE_SUCCESS - the function completed successfully
*/
mpe_Error mpeos_mediaDripFeedStop(
   mpe_MediaDecodeSession mediaDecodeSession);
```

Porting layer (native) sample code

The following code illustrates how to start, feed, and stop a video-drip feed using the porting layer functions.

```
/**********************************************************
Test Description: To test mpe_mediaDripFeedxxx methods. Test
will create and feed frames to a drip feed.

Pre-Condition: None

Actions:
• Call mpe_mediaDripFeedStart to create the drip feed
• Call mpe_mediaDripFeedRenderFrame to submit a frame for
  display
• Render several frames in a loop with a time delay in between
• Call mpe_mediaDripFeedStop to stop the drip feed
**********************************************************/

#include "common.h"

static uint32 DEFAULT_TIME_PERIOD = 5 * 1000;
   // 1 second
```

```
#define frame1_size (15800)
   // Size is in bytes
extern uint8 frame1[];
#define frame2_size (98275)
   // Size is in bytes
extern uint8 frame2[];
#define espnIFrame_size (27539)
   // Size is in bytes
extern uint8 espnIFrame[];
#define IBPFrames_size (632320)
   // Size is in bytes
extern uint8 IBPFrames[];

typedef struct _frame {
   uint8 *data;
   long dataSize;
} mpegFrame;

static mpegFrame mpegFrames[] = {
   { frame1, frame1_size },
   { frame2, frame2_size },
   { espnIFrame, espnIFrame_size },
   { IBPFrames, IBPFrames_size }
};
static ui32 numberFrames = sizeof(mpegFrames) /
   sizeof(mpegFrame);

int main(void)
{
   mpe_Error retCode = MPE_SUCCESS;
   mpe_MediaDecodeSession dripFeedSession;
   mpe_EventQueue streamEventQueue;
   mpe_MediaDripFeedRequestParams dripFeedParams;
   ui32 i;

   banner("Running - dripFeedTest");
   // Test 1 - Regular drip feed scenario
   banner("Test 1: Call drip feed start, renderFrame, and
   stop");
   // queue for stream events
   retCode = mpe_eventQueueNew( &streamEventQueue, "Drip Feed
   Test streamEventQueue" );
   if ( stat(retCode, "Queue Creation failed") )
   {
   return retCode;
   }
   // Create the drip feed decode session
   dripFeedParams.videoDevice = DEFAULT_VIDEO_DEVICE;
   retCode = mpe_mediaDripFeedStart( &dripFeedParams,
   streamEventQueue, NULL, &dripFeedSession );
   if ( stat(retCode, "mpe_mediaDripFeedStart failure") )
```

```
    {
    return retCode;
    }
    // loop reading and rendering frames
    for ( i = 0; i < numberFrames; i++ )
    {
        retCode = mpe_mediaDripFeedRenderFrame(
        dripFeedSession, mpegFrames[i].data,
        mpegFrames[i].dataSize );
        if ( stat(retCode, "DripFeedRenderFrame failed") )
        {
            break;
        }
        mpe_threadSleep(DEFAULT_TIME_PERIOD, 0); // delay so
        frame can be viewed
    }
    retCode = mpe_mediaDripFeedStop( dripFeedSession );
    stat(retCode, "mpe_mediaDripFeedStop failed\n");
    // We are done with the event queue, just delete it
    mpe_eventQueueDelete( streamEventQueue );
    banner("Exiting test");
    return retCode;
}
```

Java (JMF) additions

The following Java classes and interfaces were either added to the implementation or were updated to support video-drip feeds. The external interface class `org.dvb.media.DripFeedDataSource` was already in place and was updated to provide a functional implementation of drip feeds using the new porting layer APIs.

- `DripFeedDataSource`: A class that allows creation of a source for a JMF player to be able to feed the decoder progressively with parts of a clip (for example, I or P MPEG-2 frame) according to the drip-feed mode format defined in the MHP content format chapter (7.1.3).

- `DripFeedPlayer`: A drip feed content specific `MediaHandler` for rendering and controlling drip-feed data received from a `DripFeedDataSource`.

- `Handler (com.vidiom.impl.media.content.vidoe.dvb.mpeg.drip)`: A generic class that extends `DripFeedPlayer` and is used by the Java media manager to instantiate a drip-feed player based on a `DripFeedDataSource`.

- `DataSource (com.vidiom.impl.media.protocol.dripfeed)`: A generic class that extends `DripFeedDataSource` and is used by the Java media manager to instantiate a drip-feed data source from the URL `dripfeed://`.

- `DripFeedPresentation`: A class that handles the presentation aspects of video playback for players which includes properties such as visibility and size.

- PlayerAssociationControl: A control interface implemented by a DataSource object which provides a method to allow a player to associate itself with the DataSource.

Other existing Java classes that are used internally by the drip-feed implementation or by drip-feed applications are as follows:

- SecurityManager: A class that allows applications to implement a security policy. It allows an application to determine, before performing a possibly unsafe or sensitive operation, what the operation is and whether it is being attempted in a security context that allows the operation to be performed. The application can allow or disallow the operation.
- Manager: Is the Java media manager. An access point for obtaining system dependent resources such as Players, DataSources, and the system TimeBase.
- DripFeedPermission: A class that represents permission to access the drip-feed mode.

Java pseudo code

The following steps can be taken to create, start, and stop a drip feed based on MPEG-2 video frames stored in local files - one frame per file (see the MHP specification, Annex W.3, for full source code):

1. Set up the drip feed for playback.

```
// Create an instance of DripFeedDataSource
DripFeedDataSource dripDataSource = new
    DripFeedDataSource();

// Use the manager to create a player with an associated
drip feed
Player dripPlayer = Manager.createPlayer( dripDataSource );
    player.start();
```

2. Feeds the frames to the decoder.

```
// Retrieve MPEG-2 frames from file or network
byte[] frame;
do {
    FileInputStream fin = new FileInputStream(
    “frameXXX.mpg”);
    fin.read( frame );

    // Send MPEG-2 frames to the player via the DataSource
    dripDataSource.feed( frame );
} while ( frame != null );
```

3. Shuts down

```
player.stop()
player.close()
```

A v1.0-ER1 Release Information

Overview

This section describes the changes that were made between the v0.9.5 and v1.0-ER1 releases. It covers the following topics:

- *Build changes*
- *Known bugs*
- *Supported platforms and dependencies*
- *OCAP I16 compliance*
- *Currently supported OCAP APIs*
- *Bound application access to shared classes support*
- *com.vidiom.impl.manager.service.DVRServiceMgrImpl*
- *Consistency changes*
- *DisableMediaStorageVolume*
- *DSMCC loading*
- *DVR: Scheduled recording delay*
- *HAVi: Query Supported Events*
- *In-band PSIP in SITP/SIDB*
- *Initial monitor application (ECN913-4)*
- *mpeenv.ini configuration*
- *PAT/PMT changes*
- *StatusManager (SNMP support)*
- *Switched digital video updates*
- *Video presentation control*

Build changes

This release is being provided as an integral part of the Scientific Atlanta Axiom code release. It is not being distributed as a separately installable package at this time.

Known bugs

This section provides information about known P1 and P2 bugs.

Digital Video Recorder (DVR)

The following are the known DVR bugs:

ID	Summary
2637	`ExtendedResourceUsages`, used for scheduled recordings, references the caller context of the application which scheduled the recording. However, the scheduling application may not be around when the recording begins (because of rebooting, the application has been destroyed, etc.). This causes inconsistencies with resource management at the time the recording begins, or when other recordings are scheduled
4210	Unexpected `RecordingTerminatedEvent` generated

File system

The following is the known file system bug:

ID	Summary
5096	`LoadedFileSys` returns potentially wrong values

Graphics

The following is the known graphics bug:

ID	Summary
4703	Rectangle clipping is handled incorrectly in OpenCable™ Application Platform (OCAP) stack

Media presentation

The following are the known media presentation bugs:

ID	Summary
4709	Video turns black using OCAP Digital Navigator (ODN) guide after a minute of pausing live video
5062	`AWTVideoSizeControl.getSize()` does not account for Decoder Format Conversion (DFC)

Object carousel

The following are the known object carousel bugs:

ID	Summary
3254	There is a possibility the set-top box will be unable to read some sections in large modules on in-band object carousels
3630	When the carousel operates in a non-monitoring mode with regard to the Dynamic Invocation Interface (DII), it holds a global lock. Although we are unlikely to see a problem, there is a possibility a problem would occur if the carousel goes away, and attempts to read the carousel would cause all carousel activity to stall out for the time-out period

PAT/PMT

The following are the known Program Association Table (PAT)/Program Map Table (PMT) bugs:

ID	Summary
4266	OCAP stack does not provide time-shifted service information/PMT
4561	Out-Of-Band (OOB) PAT/PMT `TableChangeListeners` sometimes fails to pick up Packet Identification (PID) changes
5029	`ServiceContextImpl` fails to send out `NormalContentEvent` after successfully tuning to PAT/PMT transmission
5035	Failure to stop presenting stream when PMT PID for service is removed from PAT
5039	Undelivered PMT change upon PMT PID switch
5044	Picture-In-Picture (PIP) window does not properly update to PAT/PMT changes
5110	JMF not switching to live when handling a PAT/PMT change

PowerTV

The following are the known PowerTV bugs:

ID	Summary
3525	Early halt of playback before the end of recording
3549	While running WatchTV, production assert (0x92) crash
3820	Capture manager uses incorrect TVPID. When a new tune is done with the second tuner, the player or video stream associated with the `BackgroundVideoPresentationControl` gets switched to the new player/stream
3935	Assert failed in `avfs_SelectPartitionForRec` on tune when the disk is full. PowerTV does not stop ongoing recordings with enough space to keep the set-top box from crashing

ID	Summary
3952	Problem tuning to digital channels. PowerTV could not find Packet Identification (PID) information for the service
4710	Closed Captioning does not work
5074	Crashes when changing from standard-definition to high-definition programming

Section filtering

The following is the known section filtering bug:

ID	Summary
3091	Section filtering pending read list `sectionsRemaining` count is off

Service information

The following is the known service information bug:

ID	Summary
5071	Service Information Database (SIDB) status is not fully updated in time for Java the tune complete event

Switched digital video

The following is the known switched digital video bug:

ID	Summary
5080	`SwitchedDigitalVideo` - new locator does not automatically reselected (no `PresentationChangedEvent`) with `modifyService()`

Time-shift buffer (TSB)

The following are the known TSB bugs:

ID	Summary
4233	Standard definition/high-definition switching appears to shutdown the TSB
4266	Stack does not provide time-shifted service information/PMT

Tuning

The following are the known tuning bugs:

ID	Summary
5097	Deadlock during `TuneTest : app` becomes non-responsive while tuning
5082	Failure to tune and present the broadcast service whose `sourceID`, `frequency`, `programNumber`, and `Qam` are specified in the `hostapp.properties`

Supported platforms and dependencies

SA-8300

The SA-8300HD port now uses the PowerTV operating system version ADK61511_2p-Explorer8kg3-ATSC-SA-pKey.

OCAP I16 compliance

The unimplemented OCAP Engineering Change Numbers (ECN) and Engineering Change Orders (ECO) (limitations from the OCAP I16 specification (OC-SP-OCAP1.0-I16)) are as follows:

ECN/ ECO	Description
645	Provide a Vertical Blanking Interval (VBI) data retrieval function (I12)
661	Clarify a rule of video/graphics resolution
742	742 `LogicalStorageVolume` Re-creation
911	Media time tags
944	Persistent memory rules

Currently supported OCAP APIs

This section provides information about the currently supported OCAP Application Programming Interfaces (APIs).

Core Java APIs (J2ME/pJava)

The following are the core Java APIs:

Core Java APIs (J2ME/pJava)	Package Implemented?	Notes
`java.awt`	yes	
`java.awt.event`	yes	
`java.awt.image`	yes	
`java.beans`	yes	
`java.io`	yes	
`java.lang`	yes	
`java.lang.reflect`	yes	
`java.math`	yes	
`java.net`	yes	
`java.rmi`	yes	
`java.security`	yes	
`java.security.cert`	yes	

Core Java APIs (J2ME/pJava)	Package Implemented?	Notes
`java.security.spec`	yes	
`java.util`	yes	
`java.util.zip`	yes	

JMF-1.0

The following are the Java Media Framework (JMF)-1.0 APIs:

JMF-1.0 APIs	Package Implemented?	Notes
`javax.media`	partial	`Broadcast`, `TimeshiftBuffer`, and `RecordedService` controls working. Audio drip support also provided. Video drip support will be added by Scientific Atlanta post this release
`javax.media.protocol`	partial	`Broadcast`, `TimeshiftBuffer`, and `RecordedService` controls working. Audio drip support also provided. Video drip support will be added by Scientific Atlanta post this release

JSSE-1.02

The following are the Java Secure Socket Extension (JSSE)-1.02 APIs:

JSSE-1.02 APIs	Package Implemented?	Notes
`javax.net`	yes	
`javax.net.ssl`	yes	
`javax.security.cert`	yes	

JavaTV 1.0

The following are the core JavaTV 1.0 APIs:

JavaTV APIs	Package Implemented?	Notes
`javax.tv.graphics`	yes	
`javax.tv.locator`	yes	
`javax.tv.media`	yes	
`javax.tv.net`	yes	

JavaTV APIs	Package Implemented?	Notes
`javax.tv.service`	yes	Exceptions: `SIManager.registerInterest()` can be called, but has no effect. Program Event support not implemented because it is defined by Society of Cable Telecommunications Engineers (SCTE) profiles 4 and 5 which are not currently used by any head-end
`javax.tv.service.guide`	yes	Exceptions: Currently support profiles 1 and 2. Support for profiles 3, 4, 5 not implemented because no head-ends in U.S. currently provides Event Information Table (EIT) service information data
`javax.tv.service.navigation`	yes	As per OCAP Specification, `CAIdentification` should not be used by applications
`javax.tv.service.selection`	yes	
`javax.tv.service.transport`	yes	
`javax.tv.util`	yes	
`javax.tv.xlet`	yes	

DAVIC-1.4.1p9

The following are the Digital Audio Visual Council (DAVIC)-1.4.1p9 APIs:

DAVIC-1.4.1p9 APIs	Package Implemented?	Notes
`org.davic.media`	yes	
`org.davic.mpeg`	yes	
`org.davic.mpeg.sections`	yes	
`org.davic.net`	yes	
`org.davic.net.tuning`	yes	
`org.davic.resources`	yes	

MHP-1.0.3

The following are the Multimedia Home Platform (MHP)-1.0.3 APIs:

MHP-1.0.3 APIs	Package Implemented?	Notes
`org.dvb.application`	yes	
`org.dvb.dsmcc`	partial	Mostly complete. Support for streams and stream events will be provided by Scientific Atlanta post this release
`org.dvb.event`	yes	
`org.dvb.io.ixc`	yes	
`org.dvb.io.persistent`	yes	
`org.dvb.lang`	yes	
`org.dvb.media`	partial	Video drip support will be added by Scientific Atlanta post this release
`org.dvb.net`	yes	
`org.dvb.net.rc`	yes	
`org.dvb.net.tuning`	yes	
`org.dvb.test`	yes	All `DVBTest` methods throw `IOException` exceptions as this class is not currently being used for testing an OCAP stack (`org.ocap.test` is used instead)
`org.dvb.ui`	yes	
`org.dvb.user`	yes	

HAVi-1.1

The following are the Home Audio Video Interoperability (HAVi)-1.1 APIs:

HAVi-1.1 APIs	Package Implemented?	Notes
`org.havi.ui`	yes	
`org.havi.ui.event`	partial	Mostly complete. Virtual keyboard support will be supported by Scientific Atlanta post this release

OCAP-1.0 (I16)

The following are the OCAP-1.0 APIs:

OCAP-1.0 APIs	Package Implemented?	Notes
`org.ocap`	yes	
`org.ocap.application`	yes	
`org.ocap.event`	yes	
`org.ocap.hardware`	yes	
`org.ocap.hardware.pod`	yes	Stack support is complete. Port must be customized to specific platform
`org.ocap.media`	partial	Vertical Blanking Interval (VBI) (analog) closed captioning access will be added by Scientific Atlanta post this release
`org.ocap.mpeg`	yes	
`org.ocap.net`	yes	
`org.ocap.resource`	yes	
`org.ocap.service`	yes	
`org.ocap.si`	yes	
`org.ocap.storage`	yes	
`org.ocap.system`	yes	
`org.ocap.system.event`	yes	
`org.ocap.test`	yes	
`org.ocap.ui.event`	yes	

OCAP DVR I02

The following are the OCAP DVR I02 APIs:

OCAP DVR APIs	Package Implemented?	Notes
`org.ocap.dvr`	partial	Exception: Complete except recorded applications will be post v1.0
`org.ocap.shared.dvr`	partial	Exception: Complete except DVR time-line support will be added by Scientific Atlanta post this release

OCAP DVR APIs	Package Implemented?	Notes
`org.ocap.shared.dvr.navigation`	yes	
`org.ocap.dvr.storage`	partial	Exception: Complete except support for `MediaStorageVolume` class will be implemented by Scientific Atlanta post this release
`org.ocap.shared.media`	partial	Exception: Complete except DVR time-line support will be implemented by Scientific Atlanta post this release

OCAP front panel extension

The following is the OCAP front panel extension API:

OCAP front panel API	Package Implemented?	Notes
`org.ocap.hardware.frontpanel`	yes	

OCAP home networking extension

The following is the OCAP front panel extension API:

OCAP home networking API	Package Implemented?	Notes
`org.ocap.hn`	no	Post v1.0
`org.ocap.hn.content`	no	Post v1.0
`org.ocap.hn.content.navigation`	no	Post v1.0
`org.ocap.hn.services`	no	Post v1.0
`org.ocap.hn.util`	no	Post v1.0

OCAP ScaledVideoMgr extension

The following is the OCAP `ScaledVideoMgr` extension API:

OCAP scaled video API	Package Implemented?	Notes
`org.ocap.system.ScaledVideoManager`	partial	Parts of the draft ECR for this package were implemented. This package will likely be deprecated in deference to the new `MultipleScreenManager` facility being added to the core OCAP specification (still in an ECR state)

Bound application access to shared classes support

Support for ECN 852-4 has been completed. The bulk of support for ECN852 was added in v0.9.5-ER3. With this release the implementation has been updated to follow the authentication rules specified for shared classes. Changes are summarized as follows:

- Shared classes and other files stored during API registration must authenticate as dual-signed, within the context of the registering application.
- Shared classes are not authenticated at use-time against the referencing application.

com.vidiom.impl.manager.service.DVRServiceMgrImpl

The class `com.vidiom.impl.manager.service.DVRServiceMgrImpl` provides necessary support for recorded services and is required for a configuration that supports the recording and playback of recorded services. To effectively override the default `com.vidiom.impl.manager.service.ServiceMgrImpl` class, include an entry like the following in the `mpeenv.ini` file:

```
OCAP.mgrmgr.Service=com.vidiom.impl.manager.service.
    DVRServiceMgrImpl
```

NOTE: This extension of the `ServiceMgrImpl` class supports both broadcast and recorded services.

Consistency changes

Consistency changes were made to the OCAP stack in the following areas: `mpeos_gfx.h`, `mpe_gfx.h`, DirectFB, MPE memory, Hard-Disk Drive (HDD) Spin-Down Manager, and Java updates.

mpeos_gfx.h

Changes were made to the `mpeos_gfx.h` header file to make it consistent with the other APIs. The cleanup includes the following changes:

- `mpe_GfxColorFormat` was updated and enhanced:
  - changed `MPE_GFX_ARGB32` to `MPE_GFX_ARGB8888` and `MPE_GFX_ARGB16` to `MPE_GFX_ARGB1555` for consistency
  - removed unused `MPE_GFX_A8`, `MPE_GFX_A4`, and `MPE_GFX_A1`
  - added `MPE_GFX_A8RGB888` and `MPE_GFX_A2RGB565` (for future use)
- `mpe_GfxBitDepth` was updated with `MPE_GFX_2BPP` and `MPE_GFX_4BPP` for completeness
- `mpe_GfxAlphaChannel` was removed as it was not used and `mpe_GfxSurfaceInfo` updated accordingly
- `mpe_GfxCapabilities` was removed as it is no longer used

mpe_gfx.h

Removed `mpe_gfxGetCapabilities()` from the `mpe_gfx.h` header file, because it is no longer used.

DirectFB

Refactored the port-specific DirectFB functions in the SA-8300HD port to insure its local global variables are named differently than the globals in the DirectFB module to alleviate duplicate symbol warnings during disk-image creation.

MPE memory category

Refactored the MPE memory to make it consistent. The cleanup includes the following changes:

- removed the unused MPE memory categories `MPE_MEM_OPQAUE_TABLE`, `MPE_MEM_OPAQUE_FILTER`, and `MPE_MEM_GFX_FONT`
- refactored `MPE_MEM_TABLE_FILTER` and `MPE_MEM_SECTION_FILTER` to use the single memory category `MPE_MEM_FILTER`
- removed the Scientific Atlanta port-specific `MPE_MEM_SNMP` memory category and made it into a port-specific memory category extension

HDD manager

Removed the Scientific Atlanta-specific HDD Spin-Down Manager (Java and MPE/MPEOS) from the core OCAP Stack. SA will be re-factoring this code into an SA-specific extension module in the future.

Java updates

Originally, the OCAP stack was being developed using a pJava-based JVM. However, we were using a J2ME/PBP-based JVM for a while now (years), and so the old pJava-based sources within the OCAP Java tree were removed.

Unused Sun/Esmertec reference implementation sources for JavaTV and JMF were removed from the OCAP Java source tree.

EVM Updates include:

- add new `-verbose:dac` EVM logging enhancement (to log whenever the EVM compiler finds something too complicated to cache)
- exposing of the internal `jeode.evm.codeBuffers.allocBlock` and `jeode.evm.codeBufferes.compilerBufSize` properties to be set for a production build
- disabled verification of classes from trusted class-loaders (for example, classes found on the classpath) to enhance the OCAP stack boot-up performance (all OCAP xlet application classes will still go through verification)

DisableMediaStorageVolume

This release includes the implementation of the OcapRecordingManager enableBuffering() and disableBuffering() from the most recent version of ECR929. Disabling buffering causes all buffering to be terminated (including buffers supporting buffering requests and time-shifted services). Additionally, on-going recordings are suspended (and segmented) while system wide buffering is disabled.

It should be noted that the buffering state as controlled through the enableBuffering() and disableBuffer() methods and is persisted across reboots (if buffering is disabled at power down, it remains disabled when the OCAP stack is rebooted). However, the disabled state of MediaStorageVolumes (also addressed in ECR929) is not persisted in 1.0-ER1. This will be addressed with full implementation of DVR StorageManager.

DSMCC loading

Directories, streams, and stream events can now be loaded via DSMCCObject.synchronousLoad() and DSMCCObject.asynchronousLoad().

Attempts to create a DSMCCStream or DSMCCStreamEvent from a non-loaded DSMCCObject will now throw an NotLoadedException, per the OCAP specification.

DVR: Scheduled recording delay

Support is now available for scheduled recording delay at cold boot (ECN856). The former mpeenv.ini variable OCAP.dvr.recording.delayScheduleTimeout is no longer supported, and has been replaced by the OcapRecordingManager.setRecordingDelay() method as defined by ECN856.

Unlike previous versions of the OCAP-stack implementation, the recording database will be fully loaded during the specified delay period. Recordings can be scheduled, browsed, deleted, presented, etc. during the recording delay period. Only the physical recording process will be prevented during the delay period.

As in previous versions of the OCAP-stack implementation, any call to OcapRecordingManager.signalRecordingStart() prompts the recording manager to resume normal operation.

HAVi: Query Supported Events

The new HAVi implementation includes query supported events which includes `HEventRepresentation string`, `HEventRepresentation Color`, and `HEventRepresentation Symbol`.

Full support is now implemented for the specification and retrieval of `HEventRepresentation` for remote-control key events as follows:

```
HEventRepresentation her =
   HRcCapabilities.getRepresentation(HRcEvent.VK_POWER);
```

The stack introduces a new interface that returns the platform-specific `HEventRepresentation` associated with the given event code. The interface is defined as follows:

```
package com.vidiom.impl.havi;

public interface HEventRepresentationDatabase
{
public HEventRepresentation getEventRepresentation(
   int eventCode);
}
```

where

`eventCode` specifies the requested event code.

OCAP-stack ports can provide their own implementation of this interface by implementing `com.vidiom.impl.havi.port.mpe.HEventRepresentationDatabaseImpl` in the `<OCAPROOT>/target/<OCAPTC>/java` directory.

Alternatively, the stack provides a default implementation of this interface in which `HEventRepresentations` are read from a properties file specified by a new OCAP property (`OCAP.havi.eventRepresentationsFile`). The property file is loaded as a resource of `com.vidiom.impl.havi.HEventRepresentationDatabase`.

A valid event representation property file must have an entry (or entries) for each mandatory OCAP key code as described in OCAP I16 Section 25.2.1.2. An example property file would look like this:

```
VK_EXIT.color=BLUE
VK_EXIT.symbol=Images/exit.jpg
VK_EXIT.string=Exit

VK_POWER.color=RED
VK_POWER.symbol=Images/power.jpg
VK_POWER.string=Power

VK_REWIND.color=BLACK
VK_REWIND.symbol=Images/rwd.jpg
VK_REWIND.string=REW
VK_COLORED_KEY_3.supported=false
VK_NEXT_FAVORITE_CHANNEL.supported=false
.....
```

Not all remote control devices will contain all of the keys described in OCAP I16 Section 25.2.1.2. For those keys not supported by a particular device (in the example file excerpt above, `VK_COLORED_KEY_3` and `VK_NEXT_FAVORITE_CHANNEL`) you can add an entry indicating lack of support. For supported keys, it is not necessary to specify a `<em>supported</em>` property entry in the file, simply specify some combination of `//string//`, `//color//`, and `//symbol//`.

HEventRepresentation string

Event strings are required attributes for all supported events. Failure to supply an event string will result in failure to create an event representation for that event.

HEventRepresentation Color

Event colors are specified either by `java.awt.Color` constant or RGB formatted hexadecimal integer prefaced with `0x`. For example, both of these entries describe the same event representation:

```
VK_POWER.color=red
VK_POWER.symbol=/Images/power.jpg
VK_POWER.string=Power

VK_POWER.color=0xFF0000
VK_POWER.symbol=/Images/power.jpg
VK_POWER.string=Power
```

Color attributes are only required for the `VK_COLORED_KEY_?` events. For all other events, color is optional.

HEventRepresentation Symbol

Event symbols are graphical representation of a particular remote control key. Symbols are specified as Java class resources of `HEventRepresenationDatabase`. Symbol attributes are optional for all event representations.

In-band PSIP in SITP/SIDB

In-band Program and System Information Protocol (PSIP) in SITP/Service-Information Database (SIDB) changes include SITP and SIDB tasks and OCAP locator resolution.

SITP task

New functionality is added to acquire and parse the in-band (IB) Cable Virtual Channel Table (CVCT) and update the service-Information Database (SIDB) when an Out-of-Band (OOB) channel table acquisition time out occurs or after a OOB channel table acquisition failure.

SIDB task

The SIDB task includes the addition of IB PSIP storage and accessors methods. These changes include the addition of get and set methods to the internal API for CVCT data and added storage to the SIDB transport-stream structure for CVCT revisioning information.

This revisioning information is used to determine table revisioning state for table acquisition/update notification to upper layers during tuning.

OCAP locator resolution

Support is added for ECN594-3. New functionality is added to the native layer to facilitate OCAP locator resolution based on IB PSIP data in the absence of the OOB service information.

Initial monitor application (ECN913-4)

Support for the initial monitor application launch (ECN913-4) has been added. Changes are summarized as follows:

- Added new `OcapSystem.monitorConfiguringSignal()` API.
    - The initial monitor application may invoke `monitorConfiguringSignal()` prior to `monitorConfiguredSignal()` to indicate that it `//promises//` to invoke `monitorConfiguredSignal` as soon as it is ready. This has the effect of cancelling the implementation's time-out (of 5 seconds) while waiting for `monitorConfiguredSignal` to be invoked.
    - `monitorConfiguringSignal()` must be called within 5 seconds instantiation of the initial monitor application initial Xlet class, else the implementation will continue with the boot process.
- Initial monitor application re-launch support has been added such that an initial monitor application always sees a pristine operating environment when it starts up.
    - If initial monitor application exits, the boot process is reinitiated prior to re-launching the initial monitor application.
    - If initial monitor application was previously not signaled (or previously failed to launch) and the latest XAIT indicates that an initial monitor application exists, the boot process is reinitiated prior to re-launching the initial monitor application.
- Implicit application download support has been added.
    - When an XAIT-signaled application that is delivered via in-band and the object carousel is auto-started, the implementation will ensure that the application can run.
    - This involves implicitly tuning (taking an available tuner or stealing one from a current use -- without going to resource contention), downloading the application, and then releasing the tuner.

- Removed existing support for a stallable boot process (whereby the implementation would stall the launch of XAIT-signaled applications until an initial monitor application could be launched if one was signaled).

◆ **NOTE:** A change has been made compared to the previous Axiom implementations of the stack regarding ordering of `AUTOSTART`s within the initial monitor application abstract service. OCAP does not specify any guaranteed ordering for the `AUTOSTART`ing of applications signaled as `AUTOSTART` in the same abstract service. OCAP specifies that there should be no other `AUTOSTART` applications besides the initial monitor application in the initial monitor application abstract service (See OCAP-1.0 I16 Section 21). Where the previous Axiom release would delay `AUTOSTART`ing of non-initial monitor application applications until the initial monitor application has been launched, this release provides no such guarantee.

mpeenv.ini configuration

For detailed information regarding environment variables in the `mpeenv.ini` file, refer to Configuring the `mpeenv.ini` file section in the Porting Guide. The following are new or modified environment variables for this release:

`OC.DII.CHECK.DURATION=`*milliseconds*

specifies the amount of time to leave a Download Information Indication (DII) check filter in place before removing it, unless more accesses occur. Where:

milliseconds

specifies the number of milliseconds of the timer. Possible values range from `0` to `10000`. The default setting is `10000`.

For example:

```
OC.DII.CHECK.DURATION=10000
```

`OCAP.dvr.recording.delayScheduleTimeout`

is no longer supported.

`SITP.DEFAULT.FREQ.PLAN`

is no longer supported.

`SITP.OOB.DEFAULT.SI`

is no longer supported.

`SITP.OOB.DEFAULT.SI.DELAY`

is no longer supported.

`-verbose:gc,dac`

appended `dac` to the `-verbose:class,jni,gc,dac` variable to enable support for the new EVM logging option.

PAT/PMT changes

The MPEG-2 Program Association Table (PAT) and Program Map Table (PMT) are constructs for signaling the availability of multiple programs (often audio/video channels) within a single MPEG-2 transport stream multiplex (a physical channel) and the program components (streams) contained within each program. The mapping of programs and the program components within the channels is not necessarily static. The change in configuration of the cable network head-end equipment or the content received by the network head-end can result in a change to the PAT and/or PMT to accommodate the new equipment configuration and/or content makeup.

The previous OCAP stack implementation (up to v0.9.5-ER3) monitored updates to PAT and PMT tables and delivered appropriate system-information change events to applications registered as listeners. The OCAP stack itself did not respond to the change in available programs or program components.

The new implementation provides support for program and program component changes signaled via a PAT or PMT change. Any updates to PAT or PMT tables are now detected by dependent subsystems within the OCAP stack and handled appropriately.

In addition to the requirements implied by the change in the signaled PAT and PMT, the subsystem requirements include portability requirements. A change in time-shifted or recorded components is performed differently based upon platform capabilities. On platforms that support a change of recorded components on an ongoing time-shift or recording session, the component change is performed on the session and the newly-selected components are recorded in the appropriate metadata. On platforms which do not currently support a change in recorded components (for example, PowerTV), the OCAP stack now responds to a failed component change by starting a new time-shift buffer or recording with the new components.

Recording, JMF broadcast player, and JMF time-shift player subsystems are affected by this change.

Recording

An in-progress recording monitors all changes/updates to the components currently being converted. The recording responds to a change in the PMT associated with the recording service by:

1. Determining if the set of recording components needs to be changed based on the revised `ServiceDetails` and `ServiceComponents` to determine if PIDs actually changed.
2. Issuing an appropriate component change on the recording session.
3. If the component change fails, stop the current recording session and starting a new recording segment with the new components.

Since the OCAP stack uses TSBs to support recording (ECN817), recordings are also subject to time-shift, side effects caused by PAT/PMT changes. For instance, a change in the buffered components may require a new TSB to be started on platforms which cannot change the buffering components on the fly - resulting in a new `RecordedService` being added to the `SegmentedRecordedService`.

If a recording is segmented as a result of component changes or TSB shutdown, the internal state of the recording transitions to `IN_PROGRESS_WITH_ERROR` and appropriate notifications are propagated.

Recording metadata contains all relevant component information pertaining to the segment. For each recording segment, the recorded stream type (A/V/CC/PCR), associated PID returned by MPEOS, and associated language for each PID etc. is persistently stored.

JMF broadcast player

The JMF broadcast player monitors changes/updates to the components (audio, video) currently being presented based upon the selected locator(s). The player responds to a change in the PMT associated with the presenting service by:

- determining if the set of presenting PIDs needs to be changed based on the revised ServiceDetails and ServiceComponents
- issuing an appropriate component change on the decode session
- issuing appropriate notifications. For service locators or service-component locators which contain remapable components (for example, component tags), presentation should continue after the PAT/PMT change with little interruption.

JMF time-shift player

The JMF time-shift player does not directly respond to changes in the broadcast PAT and PMT since the change to the broadcast components only affects the buffering of content into the time-shift. However, the JMF time-shift player is subject to side-effects caused by PAT/PMT changes. For instance, a change in the buffered components may require a new TSB to be started on platforms which cannot change the buffering components on the fly. On PowerTV, the side-effect is more dramatic. A change in the buffered components will shutdown both buffering and playback from the current TSB . This causes the JMF time-shift player to jump to live broadcast presentation. The TSB containing the newly-signaled components will have no content to navigate back into, so all content between the presentation point and live will be unavailable for navigation or retroactive recording.

DVR: ServiceDetails clarification

ServiceDetails is implemented as defined in ECN897, with the following exceptions:

- We do not provide support for intra-segment component changes (for portability)
- We do not provide multi-language recorded service details (`MultiString` support)
- We do not provide overall completeness of `RecordedService` JavaTV system information

Support for PAT/PMT changes

The native DVR functions for buffering/playback and conversion have changed to support PAT/PMT changes.

A new DVR-specific PID data structure was defined instead of the `mpe_PID` used for broadcast streams.

As a result all APIs using `mpe_PID` are now using `mpe_DvrPidInfo` or `mpe_DvrPidTable` instead. These functions are:

```
mpeos_dvrRecordingPlayStart()
mpeos_dvrRecordingTsbStart()
mpeos_dvrTsbBufferingStart()
mpeos_dvrTsbBufferingChangePids()
```

The following functions now take `mpe_DVRPidTable`:

```
mpeos_dvrPlaybackGetPids()
mpeos_dvrPlaybackChangePids()
mpeos_dvrTsbConvertStart()
```

MPE_DVR_PID_UNKNOWN

`MPE_DVR_PID_UNKNOWN` specifies the PID is unknown and is defined as follows:

```
#define MPE_DVR_PID_UNKNOWN (-1)
```

`MPE_DVR_MAX_PIDS`

`MPE_DVR_MAX_PIDS` specifies the maximum number of PIDs allowed in the `mpe_DvrPidTable` and is defined as follows:

```
#define MPE_DVR_MAX_PIDS (10)
```

mpe_DvrPidInfo

`mpe_DvrPidInfo` specifies DVR PID definitions and is defined as follows:

```
typedef struct _mpe_DvrPidInfo {
    mpe_MediaStreamType streamType;
    int16 srcPid;
    int16 recPid;
    mpe_SiElemStreamType srcEltStreamType;
    mpe_SiElemStreamType recEltStreamType;
} mpe_DvrPidInfo;
```

where

streamType specifies the audio, video, data, PCR, etc.

srcPid specifies the requested PID.

recPid specifies the actual recorded PID.

srcEltStreamType
specifies screen elements, for example, MPEG1, MPEG2, MHEG, etc. as defined in `mpeos_si.h`.

recEltStreamType
specifies recording elements, for example, MPEG1, MPEG2, MHEG, etc. as defined in `mpeos_si.h`.

mpe_DvrPidTable

mpe_DvrPidTable specifies DVR PID table and is defined as follows:

```
typedef struct _mpe_DvrPidTable {
    int64 mediaTime;
    uint32 count;
    mpe_DvrPidInfo pids[MPE_DVR_MAX_PIDS];
} mpe_DvrPidTable;
```

where

mediaTime specifies the media time when the PIDs are set.

count specifies the number of PIDs.

pids specifies the array with the actual recorded PIDs.

mpe_MediaStreamType

mpe_MediaStreamType streamtype definition was added to mpeos_media.h and is defined as follows:

```
typedef enum _mpe_MediaStreamType {
     MPE_MEDIA_UNKNOWN,
    MPE_MEDIA_VIDEO,
    MPE_MEDIA_AUDIO,
    MPE_MEDIA_DATA,
    MPE_MEDIA_SUBTITLES,
    MPE_MEDIA_SECTIONS,
    MPE_MEDIA_PCR
} mpe_MediaStreamType;
```

where

MPE_MEDIA_UNKNOWN

specifies an unknown media type.

MPE_MEDIA_VIDEO

specifies video media.

MPE_MEDIA_AUDIO

specifies audio media.

MPE_MEDIA_DATA

specifies data.

MPE_MEDIA_SUBTITLES

specifies subtitles.

MPE_MEDIA_SECTIONS

specifies media sections.

MPE_MEDIA_PCR

specifies PCR.

Service information (PMTManagerImpl)

Added support for prioritized PMT change listeners to the package com.vidiom.impl.ocap.si.

addInBandChangeListener is for use by the OCAP implementation only. addInBandChangeListener registers an in-band TableChangeListener to the service described by the given locator. Multiple listeners registered to the same service will be notified in decreasing priority order. All listeners registered using this API will be notified on the SystemContext. addInBandChangeListener is defined a follows:

```
public class ProgramMapTableManagerImpl
 extends ProgramMapTableManager implements TableManagerExt
```

```
public void addInBandChangeListener(
    TableChangeListener listener,
    Locator locator,
    int priority )
```

where

listener	specifies the listener to be notified of in-band table changes.
locator	specifies the in-band service of interest.
priority	specifies the priority of this listener.

JMF live playback

While in live mode (that is, while decoding directly from the `NetworkInterface` and not trough a TSB) JMF watches for PAT changes (in particular, service removal) by registering a `ServiceDetailsChangeListener` on the transport object associated with the `ServiceDetails` of the service being presented by the player.

JMF watches for PMT changes by registering a `TableChangeListener` on the `ProgramMapTableManager`.

JMF time-shifted playback

In time-shift mode (that is, while decoding content from the TSB) a PAT change (in particular, Service removal) occurs when the TSB resources are taken away during playback. PMT changes are detected two ways:

- Intra-TSB changes (changes within a single TSB) are signalled by events from native during playback
- Inter-TSB changes (changes that occur because a new TSB was created when the PMT changed) are detected by hitting end-of-file during playback. In this case, JMF stops decoding the current TSB and start decoding the next TSB. You may notice a blip while the player switches between TSBs.

Currently, only the latter form of time-shifted PMT monitoring is supported.

When a PMT change occurs while time-shifted, the player will go through the process of determining which components to present, based on service information obtained from the live point.

Time-shifted service information is not yet used by JMF.

StatusManager (SNMP support)

During testing and validation of the OCAP stack various methods were devised for gathering status information from various components of the OCAP stack at both the Java and native layers. This design attempts to provide a uniform framework for the provisioning of status information. The intent is to provide a framework that allows status producers to register with a centralized component, such that any form of status consumer can gain access to status information that is both pulled from status producer and pushed to the status consumer.

The significant components of the framework that provide access to stack status information were designed within the Java level. This strategy was considered the best approach since it provides the most flexibility for how the status information is consumed and potentially delivered to other remote consumers (for example, a head-end diagnostic agent utilizing Simple Network Management Protocol (SNMP) Management Information Bases (MIB)).

StatusRegistrar

The StatusRegistrar is a centralized class that allows StatusProducers to register their services for discovery and access by status consumers. Each status producer registers by name with the StatusRegistrar. Status consumers can lookup StatusProducers by name, which allows them to gain access to the status information services of the StatusProducer. Additionally, the StatusRegistrar provides a method to acquire a complete list of all currently registered StatusProducers.

StatusProducer

The StatusProducer interface can be implemented by any component of the OCAP stack that contains information that may be of interest for purposes of debugging, profiling, or testing. A StatusProducer registers its services with the StatusRegistrar, which allows status consumers to gain access to the status producer. A StatusProducer can provide a complete list of status information types that may be provided to status consumers. Each status type is identified by a string name.

The status types provided may be either static (pulled) or active (pushed). Static information types make their associated status information available upon explicit synchronous request. Whereas the information associated with active status types is made available via asynchronous status events delivered to classes implementing the StatusListener interface.

StatusListener

The StatusListener interface is implemented by any class that wishes to function as a status consumer for asynchronous status events defined by a given StatusProducer. A StatusListener will register as a listener with a StatusProducers for one or more specific status types. When activity within the OCAP stack results in modifications or changes to the state of a particular StatusProducer the relevant StatusListener will be invoked with the requested type of status information.

MPEStatusProducer

The MPEStatusProducer is a StatusProducer that provides access to status information within the native portion of the OCAP stack. Currently, there are only a few synchronous and asynchronous status types available through the MPEStatusProducer. The status types currently defined are meant to be examples for further build out of the support for native status information. In support of the MPEStatusProducer a number of new MPE and MPEOS APIs were defined for acquiring synchronous status information and registering for delivery of asynchronous status information. The new MPE/MPEOS APIs were designed with the intent of making them as flexible and general as possible in order to make them easily extensible for various types of native status information.

Switched digital video updates

Switched digital video updates include retune support for network, object carousel, DAVIC `SectionFilter`, JMF broadcast, and DVR recordings and time-shift retune.

Network interface retune support

Added new `ServiceDetailsCallback` so sub-systems within the implementation could listen for service re-mapping. Interested observers register with the `SIManager` and are notified when any service is mapped, re-mapped, or unmapped.

Modified the `NetworkInterface` implementation to listen for re-map notifications and automatically re-tune if the service being re-mapped is the one currently tuned. When the network interface is tuned, re-tuned, or untuned, all observers (those with a registered `NetworkInterfaceCallback`) are notified.

Object carousel (ServiceDomain) retune support

The MPE file system's `mpe_fileSetStat()` has added a new value which can be set.

If the mode parameter to `mpe_fileSetStat()` is `MPE_FS_STAT_SIHANDLE`, the service-information handle used to mount the carousel will be change to the value specified in the `siHandle` field of the `mpe_FileInfo` structure passed in. The carousel attempts to move the carousel to the new service specified. If the frequency containing that service is currently tuned, the carousel immediately attempts to reconnect, otherwise it reconnects whenever a tune to that frequency takes place.

This only applies to object carousels. All other file systems will return `MPE_FS_UNSUPPORT` if called with this mode value.

All parameters of the object carousel must be identical in the new frequency. All association tags and carousel IDs must be defined in the new service. If not, the carousel will fail to reconnect correctly. PIDs do not need to be the same, but the tags must map to PIDs in the new service.

Outstanding I/O operations will fail, just as with any disconnect (tune away).

This will change a `mpe_dirSetUStat()` call in a future release of MPE.

Future attempts to unmount the carousel must use the original locator used to mount the carousel, not the locator which represents where the service was remapped. For example, if the carousel is mounted with locator `ocap://f=1234.5` and the service is remapped to frequency `5432` program, you must unmount with the original locator (`ocap://f=1234.5`), not `ocap://f=5432.1`, even though the latter represents the current location of the carousel.

DAVIC SectionFilter retune support

Attached DAVIC `SectionFilterGroups` now listen to their associated `NetworkInterface` for re-tune events. When a re-tune occurs, the filter group silently detaches from the old transport stream and re-attaches to the new transport stream.

JMF broadcast retune support

JMF players listen to the associated NetworkInterface for retune events.

When a retune occurs, the JMF player temporarily stops presentation, re-determines which components to present, and starts presentation on the re-tuned NetworkInterface. You may notice a blink (black or frozen video) when the player stops decoding until it restarts decoding.

DVR recordings and time-shift retune support

Both time-shifted content and recorded content now supports switched digital video remapping of services via the newly-introduced internal NetworkInterfaceCallback mechanism described above.

A switched digital video service remap directly affects the buffering of content into the time-shift(s) which are buffering the remapped service. During the service remap, buffering into the currently-buffering TSB is stopped and a new TSB is started when the remap is complete.

Since both recording and time-shifted playback utilize the time-shift, both functions are indirectly affected by the service remap. Time-shifted playback via the JMF time-shift player is designed to be unaffected by the service remap until the media time of the remap is encountered - where the player should switch from one TSB to the other. On PowerTV, the side-effect is more dramatic. A remap of the buffered service will shutdown both buffering and playback from the current TSB. This causes the JMF time-shift player to jump to live broadcast presentation once the remap is complete. Content prior to the remap is unavailable for navigation or retroactive recording.

Recording of a remapped switched digital video service will be stopped at the point the service remap is started. When the remap of an IN_PROGRESS RecordingRequest is complete, a second RecordedService will be created and added to the SegmentedRecordedService associated with the recording request. The state of the RecordingRequest will stay in the IN_PROGRESS state and no state notifications will be performed, as required by the Switched Digital Video ECR. If the service remap fails, the RecordingRequest will transition to the IN_PROGRESS_WITH_ERROR state and perform the associated state change notifications.

Playback of the ongoing RecordedService/SegmentedRecordedService should be unaffected by the service remap until the point of the remap is encountered - where the player will momentarily switch from one RecordedService to the other.

Video presentation control

This section provides new information regarding the functionality for the VideoPresentationControl interface, Video Presentation Control JMF, Video Presentation Control (VPC) native MPEOS, and Active Format Descriptor (AFD) and Decoder Format Conversion (DFC) events.

Overview

The functionality for the `VideoPresentationControl` interface is covered by two main modules the Java Media Framework classes used by applications and the native (platform specific) modules needed to support the Java classes on the PowerTV platform. The relevant native modules for this implementation are the media manager and the display manager. The media manager covers the bulk of the information needed by the `VideoPresentationControl` interface including information for calculating video dimensions (video sizing queries), scaling information, as well as getting/setting the clipping region. The display manager provides additional information for calculating video dimensions, in particular the Decoder Format Conversion (DFC) used for determining letter box/pillar regions.

The general approach is for the JMF layer to use the native layer to retrieve the required information to implement any of the `VideoPresentationControl` methods. In some cases this may just be a thin pass through implementation to return values provided by the native layer. In other cases, the JMF layer will be required to retrieve values from the native layer, perform calculations, and finally convert to units expected by Java clients.

◆ **NOTE:** All methods in `VideoPresentationControl` can be applied to either video device (background or component window). When applied to a background player, applications will typically use the `BackgroundVideoPresentationControl` interface. It is possible that some platforms may have different support for component vs. background devices.

VPC JMF

The JMF implementation supports all methods of the VPC interface. The VPC interface can be retrieved from a JMF player object as a control and subsequently used to query for clipping, scaling, positioning, and video size information. The supported methods are as follows:

`HScreenRectangle getActiveVideoArea()`

returns the size and location of the active video area.

`HScreenRectangle getActiveVideoAreaOnScreen()`

returns the size and location of the active video area on-screen.

`Rectangle getClipRegion()`

returns the area of the decoded video that will be displayed.

`float[] getHorizontalScalingFactors()`

determines the supported discrete horizontal scaling factors in case arbitrary horizontal scaling is not supported.

`Dimension getInputVideoSize()`

returns the dimensions of the video before any scaling has taken place (but after ETR154 up-sampling).

`byte getPositioningCapability()`

determines how the video can be positioned on screen.

`HScreenRectangle getTotalVideoArea()`
: returns a relative size and location of the total video area, including any bars used for letterboxing or pillarboxing that are included in the broadcast stream, but excluding any bars introduced as a result of video filtering.

`HScreenRectangle getTotalVideoAreaOnScreen()`
: returns a relative size and location of the total video area on-screen, including any bars used for letterboxing or pillarboxing that are included in the broadcast stream, but excluding any bars introduced as a result of video filtering.

`float[] getVerticalScalingFactors()`
: determines the supported discrete vertical scaling factors in case arbitrary vertical scaling is not supported.

`Dimension getVideoSize()`
: returns the size of the decoded video as it is being presented to the user.

`Rectangle setClipRegion(Rectangle clipRect)`
: sets the region of the decoded video that will be displayed.

`float[] supportsArbitraryHorizontalScaling()`
: determines whether arbitrary horizontal scaling is supported for the currently playing video.

`float[] supportsArbitraryVerticalScaling()`
: determines whether arbitrary vertical scaling is supported for the currently playing video.

`boolean supportsClipping()`
: tests if the decoder supports clipping.

VPC native MPEOS

The MPEOS native layer provides several support routines that are used by the VPC Java layer. The following methods are used to query the platform for the necessary information to support VPC:

mpeos_mediaSetBounds()

`mpeos_mediaSetBounds()` sets the clipping region (`srcRect`) and the scaled destination output (`destRect`) and is defined as follows:

```
mpe_Error mpeos_mediaSetBounds (
    mpe_DispDevice videoDevice,
    mpe_MediaRectangle * srcRect,
    mpe_MediaRectangle * destRect )
```

where

videoDevice specifies the video device.

srcRect specifies the clipping region.

destRect specifies the scaled destination output.

mpeos_mediaGetBounds()

mpeos_mediaGetBounds() retrieves the clipping region (srcRect) and the scaled destination output (destRect) and is defined as follows:

```
mpe_Error mpeos_mediaGetBounds (
    mpe_DispDevice videoDevice,
    mpe_MediaRectangle * srcRect,
    mpe_MediaRectangle * destRect )
```

where

videoDevice specifies the video device.

srcRect is a pointer to the clipping region.

destRect is a pointer to the scaled destination output.

mpeos_mediaGetAspectRatio()

mpeos_mediaGetAspectRatio() retrieves the aspect ratio of the incoming video (MPEG aspect ratio) and is defined as follows:

```
mpe_Error mpeos_mediaGetAspectRatio(
    mpe_DispDevice decoder,
    mpe_MediaAspectRatio * ar )
```

where

decoder specifies the decoder.

ar is a pointer to the aspect ratio.

mpeos_mediaGetAFD()

mpeos_mediaGetAFD() retrieves the Active Format Descriptor (AFD) for the current video stream and is defined as follows:

```
mpe_Error mpeos_mediaGetAFD(
    mpe_DispDevice decoder,
    mpe_MediaActiveFormatDescription * fd)
```

where

decoder specifies the decoder.

fd is a pointer to the active format descriptor.

mpeos_mediaGetScaling()

`mpeos_mediaGetScaling()` retrieves information about platform support for positioning, scaling, clipping, and component video support (PIP).

```
mpe_Error mpeos_mediaGetScaling(
    mpe_DispDevice decoder,
    mpe_MediaPositioningCapabilities * positioning,
    float ** horiz,
    float ** vert,
    mpe_Bool * hRange,
    mpe_Bool * vRange,
    mpe_Bool * canClip,
    mpe_Bool * supportsComponent )
```

where

decoder specifies the target decoder

positioning is a pointer to the positioning returned value that specifies how the video can be positioned on the screen.

horiz is a pointer to the discrete list of available horizontal scaling factors.

vert is a pointer to the discrete list of available vertical scaling factors.

hRange is an output pointer to the returned indicator to specify the form of the horizontal argument.

vRange is an output pointer to the returned indicator to specify the form of the vertical argument.

canClip is a pointer returned value that indicates if the decoder supports clipping.

supportsComponent
is a pointer to the returned value that indicates if the decoder supports scaling.

mpeos_mediaGetInputVideoSize()

`mpeos_mediaGetInputVideoSize()` retrieves the input video size (before clipping and scaling) are applied and is defined as follows:

```
mpe_Error mpeos_mediaGetInputVideoSize(
    mpe_DispDevice dev,
    mpe_GfxDimensions * dim )
```

where

dev specifies the target device.

dim is a pointer to the input video size.

mpeos_dispGetDFC()

a retrieves the Decoder Format Conversion (DFC) as dictated by the client and is defined as follows:

```
mpe_Error mpeos_dispGetDFC(
   mpe_DispDevice decoder, m
   pe_DispDfcAction * action )
```

where

decoder specifies the target decoder.

action is a pointer to the DFC.

AFD and DFC events

Active Format Description (AFD) and Decoder Format Conversion (DFC) events are now supported by the native layer and are returned to Java applications as the following events:

`public class ActiveFormatDescriptionChangedEvent`

provides event signalling when the transmitted active format definition has changed

`public class DFCChangedEvent`

provides event signalling when the decoder format conversion being used has changed

The AFD information from the MPEG stream is signaled by PowerTV and then mapped to OCAP constants. This mapping covers all AFDs, but only the standard set has been fully tested. The following AFDs are all supported and tested:

```
AFD_16_9                          AFD_16_9_TOP
AFD_4_3                           AFD_SAME
AFD_NOT_PRESENT
```

The following AFDs are supported but have not been tested:

```
AFD_14_9                          AFD_14_9_TOP
AFD_GT_16_9                       AFD_4_3_SP_14_9
AFD_16_9_SP_14_9                  AFD_16_9_SP_4_3
```

The following DFC modes are not supported on this PowerTV implementation:

```
DFC_LB_14_9                       DFC_LB_2_21_1_ON_4_3
DFC_LB_2_21_1_ON_16_9
```

PowerTV supports scaling of the main video as well as the video in a PIP window. However, only one of these two windows can be scaled at the same time.