

Superseded by a Subsequent Version of Document

OpenCable™ Application Platform Specification

OCAP Digital Video Recorder (DVR)

OC-SP-OCAP-DVR-I01-040524

ISSUED SPECIFICATION

Notice

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

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

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

© Copyright 2004 Cable Television Laboratories, Inc. All rights reserved.

Document Status Sheet

Document Control Number:	OC-SP-OCAP-DVR-I01-040524			
Document Series:	OpenCable™ Application Platform Specification			
Reference:	OCAP Digital Video Recorder (DVR)			
Revision History:	I01-040524, Issued Specification			
Date:	May 24, 2004			
Status Code:	Work in- Process	Draft	Issued	Closed
Distribution Restrictions:	Author Only	GL/Member	GL/Member/Vendor	Public

Key to Document Status Codes

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

Trademarks:

DOCSIS®, eDOCSIS™, PacketCable™, CableHome™, OpenCable™, CableCARD™, CableOffice™, and CableLabs® are trademarks of Cable Television Laboratories, Inc.

Table of Contents

1 SCOPE	1
1.1 OCAP DVR PURPOSE	1
1.2 OCAP DVR REQUIREMENTS	1
1.3 OCAP DVR APPLICATION AREAS (INFORMATIVE)	1
1.3.1 Personal Video Recorder (PVR).....	1
1.3.2 Time Shift.....	2
1.3.3 Pushed Content.....	2
2 REFERENCES.....	3
2.1 NORMATIVE REFERENCES	3
3 GLOSSARY	5
4 ACRONYMS.....	7
5 CONVENTIONS.....	9
5.1 SPECIFICATION LANGUAGE	9
5.2 ORGANIZATION.....	9
6 DVR.....	11
6.1 INTRODUCTION	11
6.2 RECORDING	11
6.2.1 Recording types	11
6.2.2 Recordings.....	12
6.2.3 Lifespan of recordings.....	13
6.2.4 RecordedService	13
6.3 LISTING AND PLAYBACK	13
6.4 TIME-SHIFT BUFFER	14
6.5 RESOURCE MANAGEMENT	15
6.5.1 Resource Reservation for Future Activities.....	16
6.5.2 ResourceUsage to Represent Activities Requiring Resources.....	16
6.5.3 Resource Management of Recording Types (informative)	17
6.5.4 Storage.....	17
6.6 SECURITY	18
6.6.1 Permissions.....	18
6.6.2 Access scope of recordings.....	19
6.6.3 Content Protection.....	19
6.6.4 Minimum Security Constraints	19
6.7 OCAP DVR API.....	19
6.7.1 Additions to OCAP 1.0 Packages	19
6.7.2 OCAP DVR Packages.....	20

7	MINIMUM PLATFORM CAPABILITIES	21
7.1	BIT RATE	21
7.2	PLAYBACK RATES	21
7.3	OPENCABLE SET-TOP TERMINAL CORE REQUIREMENTS	21
ANNEX A	ORG.OCAP.DVR	23
ANNEX B	ORG.OCAP.DVR.NAVIGATION	55
ANNEX C	ORG.OCAP.MEDIA	69
ANNEX D	ORG.OCAP.STORAGE.....	71

List of Figures

FIGURE 6-1: RECORDING SESSION STATE DIAGRAM..... 13

This page intentionally left blank.

Summary

This document defines a minimal profile specification for a Digital Video Recorder (DVR) software environment for digital cable receivers with local storage. This platform is a modular extension to the OpenCable Application Platform 1.0 (OCAP 1.0). The OCAP 1.0 Specification, and the OCAP DVR specification, were developed by Cable Television Laboratories, Inc. (CableLabs) in conjunction with representatives from a number of its member cable operating companies, as well as leading software firms.

The OCAP DVR Specification is based on the OCAP 1.0 Specification and includes that Specification in its entirety.

Purpose

OCAP DVR is an application profile that includes all required Application Program Interfaces (APIs), content and data formats, and protocols up to the application level. Applications developed to the OCAP DVR Specification will be executed on OpenCable-compliant host devices. The OCAP DVR Specification allows cable operators to deploy their applications and services on all OpenCable-compliant host devices connected to their networks.

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

This page intentionally left blank.

1 Scope

This document defines a minimal profile specification for a Digital Video Recorder (DVR) software environment for digital cable receivers with local storage. This platform is a modular extension to the OpenCable Application Platform 1.0 (OCAP 1.0). The OCAP 1.0 Specification, and the OCAP DVR Specification, were developed by Cable Television Laboratories, Inc. (CableLabs) in conjunction with representatives from a number of its member cable operating companies, as well as leading software and hardware firms.

The OCAP DVR Specification is based on the OCAP 1.0 Specification and includes that Specification in its entirety.

1.1 OCAP DVR Purpose

OCAP DVR is an application interface that includes all required Application Program Interfaces (APIs), content and data formats, and protocols up to the application level. Applications developed to the OCAP DVR Specification will be executed on OpenCable-compliant host devices. The OCAP DVR Specification allows cable operators to deploy their applications and services on all OpenCable-compliant host devices connected to their networks.

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

1.2 OCAP DVR Requirements

The OCAP DVR platform has been designed to meet specific requirements that are not commonly applied to other DVR environments. Some of these requirements are related to content protection obligations that cable operators face, to the fact that broadcast event descriptions might be maintained by applications, and to the fact that the platform supports DVR applications deployed by different service providers that have no a priori knowledge of each other and therefore may compete for resources.

1.3 OCAP DVR Application Areas (informative)

The information in this section is informative to the OCAP DVR Specification.

This section identifies the applications and services that could be made available to the viewer when using an OCAP DVR-compliant terminal. The descriptions of the applications are intended to demonstrate the scope of services required from OCAP 1.0.

1.3.1 Personal Video Recorder (PVR)

A critical application enabled by this platform is a PVR. A PVR application may be an extension to an Electronic Program Guide (EPG) that enables viewers to select programming events to record, and to select recorded events for viewing. Event listing and selection may be integrated into the EPG. Future broadcast events may be selected for recording, for instance, on Thursday night a viewer might choose to record Sunday's broadcast of '60 minutes'. Once the recording is scheduled, the PVR application will record the event without further input from the viewer. A PVR might also allow users to schedule the recording of a package of events, such as a season of 'Friends'.

1.3.2 Time Shift

Another critical application can be called ‘time shift’. This set of features allows viewers to record the currently selected broadcast event, pause the current event, and rewind the current event. This capability is accomplished by the use of a ‘time-shift buffer’, which temporarily stores the broadcast stream. If a user selects to pause the live broadcast, the current frame of video is displayed, while the ongoing broadcast is still saved to the time-shift buffer. The contents of the buffer, up to the beginning of the current event, can be saved in permanent storage, and the contents of the time-shift buffer can be viewed with ‘trick-play’ modes, such as rewind and fast-forward.

1.3.3 Pushed Content

Applications may schedule recording or perform immediate recording of broadcast events with initiation by a viewer. Many services and applications may use this capability to provide promotional material or targeted ads to viewers. These applications use the same API features as PVR applications.

2 References

This section provides the normative and informative references used to create this specification.

2.1 Normative References

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

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

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

The following provisions apply to particular sources of documents in Table 2-1, Note column:

[1] CableLabs specifications are available from: <http://www.opencable.com/specifications/>

Table 2-1 lists the normative references for this specification:

Table 2-1 OCAP DVR Normative References

OCAP 1.0 Index	References	Edition	Description	Note
[1]	OC-SP-SEC-I04-040402		OpenCable System Security Specification	[1]
[2]	OCAP 1.0	OC-SP-OCAP1.0-110-040305	OpenCable Application Platform Specification, OCAP 1.0 Profile	[1]
[3]	OC-SP-HOST-CFR-116-040402		OpenCable Host Device Core Functional Requirements	[1]
[4]	FIPS-46-3	99 Oct 25	Data Encryption Standard (DES)	
[5]	FIPS-140-2	01 May 25	Security Requirements for Cryptographic Modules	
[6]	FIPS-197	2001 November 26	Advanced Encryption Standard (AES)	

This page intentionally left blank.

3 Glossary

The following definitions are used in this specification:

Personal Video Recorder (PVR) An application that enables viewers to schedule the recording of broadcast events, and to view and display previously recorded events.

Digital Video Recorder (DVR) A hardware/software platform that enables viewers to store digital video content. DVR systems enable applications such as PVRs.

Time-Shift A set of functionality that enables viewers to record, pause, and rewind/fast-forward through a real-time broadcast event.

Time-Shift Buffer A portion of memory that enables Time-Shift functionality.

Transrating A process of modifying MPEG compression to achieve greater compression.

This page intentionally left blank.

4 Acronyms

The following acronyms are used in this specification:

AVFS Audio/Visual File System

DVR Digital Video Recorder

GPFS General Purpose File System

PVR Personal Video Recorder

This page intentionally left blank.

5 Conventions

The following conventions are used in this manual:

- The following font type is used to indicate code examples, names of properties, and other information that **MUST** be entered exactly as-is: `code example font`
- **Boldfaced** text is used as emphasis.
- *Italicized* text is used to indicate section titles when referenced within the text.

5.1 Specification Language

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

MUST/SHALL This word, or the adjective **REQUIRED**, means that the item is an absolute requirement of this specification.

MUST NOT/SHALL NOT

This phrase means that the item is an absolute prohibition of this specification.

SHOULD This word, or the adjective **RECOMMENDED**, means that valid reasons may exist in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.

SHOULD NOT This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

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

Other text is descriptive or explanatory.

5.2 Organization

This document uses the OpenCable Application Platform Specification, OCAP 1.0 Profile [2] as its base. Where applicable, OCAP DVR sections reference the corresponding section within OCAP 1.0 [2].

The `org.ocap.dvr` API package is defined in Annex A, *org.ocap.dvr*. This specification also defines extensions to `org.ocap.media` and `org.ocap.resource` packages, which are defined in Annex B, *org.ocap.dvr.navigation* and Annex C, *org.ocap.media*.

This page intentionally left blank.

6 DVR

This chapter describes the OCAP DVR platform. The OCAP DVR APIs are defined and described in more detail in Annex A, *org.ocap.dvr*, Annex B, *org.ocap.dvr.navigation*, Annex C, *org.ocap.media*, and Annex D, *org.ocap.storage*.

6.1 Introduction

This specification fully defines a DVR extension to OCAP 1.0 [2]. Implementers of this specification SHALL also fully implement OCAP 1.0 [2]. OCAP 1.0 [2] and this DVR platform are related in such a way that OCAP 1.0 [2] implementations have no build-time or runtime dependencies on the DVR platform, while OCAP DVR implementations depend on the full implementation of OCAP 1.0 [2]. This specification defines a platform to enable applications to record and playback broadcast events, and to time-shift broadcast events. These functions are considered basic capabilities of a DVR platform, and the platform is limited in scope to support these basic functions. Advanced functions, such as preference engines or targeted content, are not explicitly supported by this platform and are considered application level functions. This Chapter describes the platform in a textual format, and places normative requirements on implementations. Annex A defines the platform API, and contains detailed API descriptions and normative statements.

6.2 Recording

The DVR APIs enable applications to schedule a recording, record the current event in a time-shift buffer, and record broadcast events in real-time. Recording of a currently broadcasting service MAY also include the portion of the broadcast event that is in a time-shift buffer. Applications MAY add broadcast events to a recording list maintained by the implementation, and the implementation is responsible for performing the recording operation (see Section 6.2.2, *Recordings on page 12*). Applications create recordings by calling the `record()` method of the `RecordingManager` object. This operation in effect creates an entry in the recording database maintained by the implementation. Applications control the scope of viewing and playback of recordings it has initiated, by indicating whether any other application, applications from the same service provider, or only itself can access a recording. Event descriptions are associated with recordings by the recording application, and visibility of such data is subject to the same accessibility rules applied to recordings (see Annex 6.3, *Listing and Playback*). Applications MAY pass event descriptions in a private format to the platform, but the platform SHALL NOT store its own event description data associated with a recording. Applications MAY keep event information private by maintaining a database of descriptive information separate from recordings, perhaps simply storing a key value with a recording.

6.2.1 Recording types

This section describes usage scenarios of the recording features defined by this platform.

6.2.1.1 Scheduled Viewer Recording

A viewer MAY schedule recording of a future broadcast event from a GUI, such as the EPG. The application presenting the GUI adds a `RecordingListEntry` to a `RecordList` (see Annex A, *org.ocap.dvr* for the API definition). The implementation SHALL indicate whether the scheduled recording will result in a resource contention.

6.2.1.2 Scheduled Service Recording

An application MAY schedule recording of promotional material, infomercials, trailers, etc., without being directed to do so by viewer input. The means for scheduling the recording and detecting resource contention is identical to scheduled viewer recordings.

6.2.1.3 Instantaneous Time-Shift Buffer Recording

A viewer MAY record the current broadcast event during the broadcast, via a GUI. The application presenting the GUI MAY issue a time-shift buffer record command, and sets the start and end times for the recording. As examples, an application MAY indicate that the entire contents of a time-shift buffer be copied into storage, and that ongoing broadcast be recorded, or an application MAY delineate the start and end points of the current broadcast event.

6.2.2 Recordings

The term ‘recording’ is used here to refer to a layman’s concept of a scheduled or recorded event, and does not refer to an actual Java object. This section describes how recordings are created, managed, and instantiated as Java constructs. The implementation maintains a database of recordings, represented by the `RecordingManager` singleton. Applications MAY add recordings to the database, and MAY acquire a set of recordings, represented by a `RecordingList`. A recording is represented by a compound object, which includes a `RecordingListEntry`, a `RecordingSession`, and a `RecordedService`. A `RecordingListEntry` provides the entry point to a recording, and `RecordingListEntries` are acquired by obtaining a `RecordingList` from the `RecordingManager`. Applications MAY make use of filters when obtaining a `RecordingList`, to limit the list to a qualified subset of recordings. Several filters are defined by the platform, and applications MAY create their own filters. A `RecordingListEntry` enables applications to store descriptive data accompanying a recording, and provides access to an associated `RecordingSession`. A `RecordingSession` is created whenever the `record()` method is invoked, and SHALL inhabit one of several states, such as ‘completed’, ‘failed’, ‘in progress insufficient space’, ‘in progress’, ‘incomplete’, ‘pending with conflict’, or ‘pending no conflict’. A `RecordingSession` represents a scheduled recording, and provides information about where it is stored and how much space it takes up, as well as other state information. It also provides access to a `RecordedService`, which is the means by which the recording is played back.

Implementations SHALL support the recording of data in OCAP supported formats (see OCAP 1.0 Section 9 Content Formats).

Implementations SHALL record in an implementation specific fashion any information needed to playback any video and audio streams selected for recording, as well as Closed Captioning, and Content Advisory.

The following diagram illustrates transitions between `RecordingSession` states.

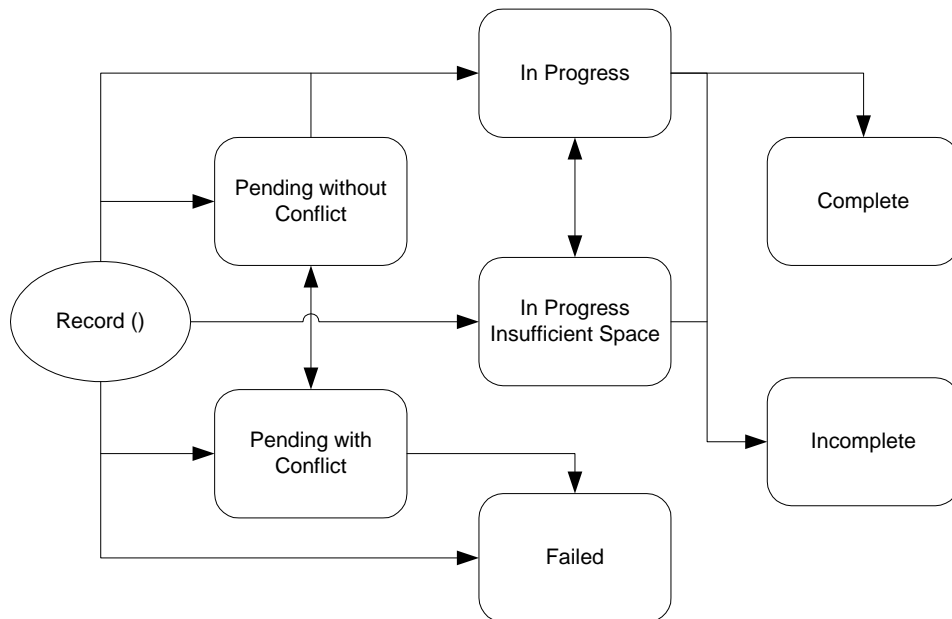


Figure 6-1: Recording Session state diagram

6.2.3 Lifespan of recordings

Applications indicate a lifespan for a recording at the time the recording is added to the recording database. Recordings can have an unlimited lifespan, or a defined lifespan. When the lifespan of a recording has been reached, the implementation SHALL delete the recorded asset and any associated files from storage and remove the recording from the recording database.

6.2.4 RecordedService

A `RecordedService` is created when a `RecordingSession` enters one of the 'in progress' states. `RecordedService` extends the JavaTV `Service` interface, and allows an application to initiate a playback using `.select(RecordedService)`. A `RecordedService` SHALL NOT be accessible through the `SIManager` class in the JavaTV service package. The `getLocator()` method of `RecordedService` SHALL return an `OCAP Locator` which is the originating service `Locator`. The `getServiceName()` method for an instance of this `Locator` SHALL return a unique `serviceName` starting with the string "RecordedService". The method `getServiceType` will return the service type of the originating service. The method `hasMultipleInstance` SHALL always return false. The implementation SHALL support the `retrieveDetails` method. All method in objects implementing `ServiceDetails` for `RecordedServices` should function the same as objects implementing the `ServiceDetail` for broadcast services.

6.3 Listing and Playback

The DVR APIs enable applications to query the implementation for a list of recordings, and to select a recording for playback. Applications MAY apply filters to qualify a request for a recording list, and the implementation SHALL not add a recording to a recording list if the requesting application is not granted listing privileges by the application that initiated the recording (see Section 6.7.2, *OCAP DVR Packages on page 20* for a discussion of the access scope of recordings). Access to recordings is controlled through file permissions, not by an explicit grant

to individual applications. An application with the monitor permission “storage” MAY override the file permission corresponding to a recording session. Recorded events in a recording list SHALL inhabit one of several states, such as pending, failed, recorded, and so on. Playback of a recording is performed by calling the `select()` method of a `ServiceContext` associated with the recording, and playback is controlled by the media controller. The media controller provides the ability for applications to vary the speed of playback, both forward and in reverse, to pause, and to skip to any point in the recording.

The following rules apply to a player associated with a service context presenting a recorded service:

- The method `getNanoseconds()` corresponding to the Media Time at the beginning of a recorded program shall return 0;
- The method `setMediaTime` called with a value of Time corresponding to `POSITIVE_INFINITY` shall set the playback location to the current record point if the recording is still on-going, or to the end of the recording.
- The method `setRate(0.0)` shall pause the playback displaying the last frame displayed. Any following `setRate` called with a non-zero parameter shall resume the playback at the specified rate from the paused location.
- Audio shall not be presented for any playback rate other than 1.0
- If recording is ongoing, and if the playback (at a rate more than 1.0) hits the end of file, the implementation shall set the playback rate to 1.0 and send a `RateChangeEvent` to any registered controller listeners. If the recording is not ongoing, the implementation shall generate an `EndOfMediaEvent`
- If the player hits the beginning of media during playback at a rate less than 0.0 the implementation shall change the playback rate to 1.0 and send a `RateChangeEvent` to any registered controller listeners.

Implementations SHALL enable simultaneous recording and playback to/from a given storage devices, when supported by corresponding storage devices. At least one storage device provided by the platform SHALL support simultaneous recording and playback. Implementations SHALL enable recording to one device, and playback from any other if multiple storage devices are present.

Closed Caption and Content Advisory data as originally delivered from the network SHALL be present on playback.

If implementations record elementary streams containing DSMCC object carousels or Application Information Tables then these shall be ignored during playback. Future versions of this platform will support playback of service bound OCAP-J applications.

6.4 Time-Shift Buffer

A time-shift buffer is used to store a finite amount of live broadcast in order to apply trick-mode controls and instantaneous record to a presenting service. Implementations SHALL support at least one time-shift buffer. The implementation SHALL store any currently selected broadcast service in a time-shift buffer. If a Host device supports simultaneous presentation of services, the implementation MAY support multiple time-shift buffers. If multiple time-shift buffers are supported, the implementation MAY support allocation of time-shift buffers and attachment of time-shift buffers to various service contexts. A time-shift buffer size MAY be changed at any time, if space is available. Each time-shift buffer is assigned by the implementation to a storage device via an `org.ocap.storage.StorageProxy` object.

When a time-shift buffer is attached to a presenting service context, the currently selected service is simultaneously decoded and displayed and stored in the time-shift buffer. A time-shift buffer is circular, which is to say that the write point wraps to the physical start of the buffer when the physical end of the buffer is reached. After boot up, or after a time-shift buffer has been flushed, A/V content is stored in the buffer at the current write point, and the buffer contains only the content that has been viewed since the boot or flush event. Once the buffer is full, the write point crosses over the original write point, and the currently presenting A/V content writes over the content stored in the buffer. A player associated with a service context that is attached to a time-shift buffer and is presenting a broadcast service shall follow all rules for a player associated with a service context presenting a recorded service. In addition the following rules also apply to such a player:

- The method `getControls()` shall return at least one object implementing the `org.ocap.media.TimeShiftControl`
- If the write point of the time-shift buffer is about to overwrite the location corresponding to the current media time due to circular buffer reaching its depth, the implementation shall set the playback rate to 1.0 in order to prevent the location corresponding the current media time being invalidated. The implementation shall send a `RateChangeEvent` to any registered controller listeners. This should occur only when the playback is at a rate less than 1.0. Timeshift buffer implementation should make sure that the location corresponding to the current media time is never invalidated.
- When playing back live broadcast, stream information will be taken directly from a decoder rather than a time-shift buffer.

Implementations MAY include 2 or more tuners, with a time-shift buffer associated with each. When the player type for a service context associated with an attached time-shift buffer, or an ongoing recording, is set to `PLAYER_TYPE_NONE`, the recording shall continue. In this case any `ServiceMediaHandlers` associated with the service context shall be removed. The service shall still be in the presenting state. If the recording is terminated on the service context a presentation terminated event is generated and the service context shall transition to not-presenting state. When the service context with `PLAYER_TYPE_NONE` is still in the presenting state and if the player type is set to background or component, the player shall start play-back from the live point

6.5 Resource Management

Resource reservation and contention management is perhaps the most complex aspect of a DVR platform. OCAP 1.0 [2] defines a resource management system, and this DVR platform complements and extends that system. Much of the control of resource allocation and contention resolution is left to applications, such as the privileged `ResourceContentionHandler` discussed below. OCAP 1.0 [2] incorporates the DAVIC resource model in which defined scarce resources are subject to a management protocol. Briefly, the protocol states that when an application requests a defined scarce resource that is in use by another application(s), the application(s) that has use of the resource is requested to relinquish the resource. If an application that has access refuses to relinquish the resource, and the requesting applications insists on gaining access, a resource contention handler may be called to resolve the conflict. A resource contention handler is an application with special privileges. If no resource contention handler is installed, the application with the highest priority is granted the resource. (See OCAP 1.0 [2] Chapter 19, “Resource Management”, for more details).

The Resource Management specified by the DVR API includes the following extensions to OCAP resource management.

6.5.1 Resource Reservation for Future Activities

The DVR platform extends the OCAP resource contention handling to include advance resource reservation for future activities in addition to the resource allocation for current activities as specified by OCAP 1.0 [2]. When an application invokes one of the `record()` methods of the `RecordingManager`, the implementation SHALL check for availability of resources for the requested record activity. If the record request is for a recording in the future, the implementation SHALL check to make sure that there are sufficient resources available to perform all other record activities scheduled for the same time period. If the record request is for a currently broadcasting event, the implementation SHALL check to make sure that sufficient resources are available to perform the newly requested record activity, all scheduled record activities that overlap the newly requested activity, and all other non-recording activities that are taking up resources. If the implementation determines that resources are not sufficient to perform the newly requested record activity, the implementation SHALL identify this as a resource conflict and use the following rules to resolve the conflict:

- If a `ResourceContentionHandler` is registered with the `ResourceManager`, the implementation SHALL invoke the method `resolveResourceContention(ResourceUsage newRequest, ResourceUsage[] currentReservations)`. The parameter `newRequest` should be an instance of `ResourceUsage` that represents the newly requested record activity. The parameter `currentReservations` should be an array of `ResourceUsage` objects. This array may contain instances of `RecordingResourceUsage` representing `RecordingSession` objects with durations overlapping the newly requested record activity and instances of `ResourceUsage` representing non-recording in-progress activities. If a `ResourceUsage` represents resources required for a `RecordingSession`, the `ResourceUsage` should also be an instance of `RecordingResourceUsage`. The implementation SHALL use the prioritized array of `ResourceUsage` objects returned by `resolveResourceContention()` to resolve the conflict. The implementation SHALL resolve the conflict in favor of activities represented by the `ResourceUsage` objects at the beginning of the array. If an activity represented by a `ResourceUsage` object failed to gain/retain reservation to a resource, the implementation SHALL keep the corresponding `RecordingSession` in `PENDING_WITH_CONFLICT_STATE`. If an activity represented by a `ResourceUsage` successfully gained reservation for required resources, the implementation SHALL keep the corresponding `RecordingSession` in `PENDING_NO_CONFLICT_STATE`.
- If a `ResourceContentionHandler` is not registered with the `ResourceManger`, the implementation SHALL use application priority to resolve the conflict.

If the implementation detects a resource conflict when the application invokes a `record()` method with `priorityFlag` set as `RECORD_IF_NO_CONFLICTS`, the `ResourceContentionHandler` SHALL NOT be invoked. The newly inserted `RecordSession` SHALL be kept in `PENDING_WITH_CONFLICT_STATE`.

If a `RecordingSession` with `PriorityFlag` `RECORD_IF_NO_CONFLICTS` conflicts with a new record request, the implementation SHALL change the state of the `RecordingSession` to `PENDING_WITH_CONFLICT_STATE`. If the conflict is thus resolved, the contention handler SHALL not be invoked.

6.5.2 ResourceUsage to Represent Activities Requiring Resources

When the implementation detects resource conflicts in a DVR platform, the implementation SHALL invoke the `resolveResourceContention(ResourceUsage, ResourceUsage[])` method instead of the `resolveResourceContention(AppID, AppID[], String)` method specified by OCAP 1.0 [2]. The implementation SHALL use an instance of `ResourceUsage` to represent all resources required for a single activity. The implementation SHALL use an instance of `RecordingResourceUsage` to represent a `RecordingSession`.

6.5.3 Resource Management of Recording Types (informative)

This section describes resource management strategies that applications MAY employ for different usage scenarios.

Scheduled viewer recording To resolve a resource contention, the GUI may notice that the program can be recorded at a different time and/or channel and offer this information in a conflict resolution UI. In this case the recording priority MAY be inverted so that lower-priority recordings take precedence.

Scheduled service recording To resolve a resource contention, if the recording application recognizes that the event can be recorded at a different time, it MAY set the recording priority lower.

Instantaneous time-shift buffer recording A viewer MAY indicate a wish to record the current broadcast event during the broadcast. The viewer might not be present after indicating the wish to record. To resolve a resource contention, a dialogue UI with time-out might be used. If the UI times out, the platform might leave this as a high-priority recording until the initial program is recorded, then invert the priority so that other recordings take precedence. The duration for this recording could be set to the duration of the initial program or a maximum setting.

6.5.4 Storage

Storage devices are represented in OCAP by objects implementing the `org.ocap.storage.StorageProxy` interface. A `StorageProxy` may be extended by `StorageOption` interfaces that expose additional capabilities of a device. The `StorageProxy` interface also supports `LogicalStorageVolumes` which allow applications to organize and control access to content. This specification extends the base storage capabilities in OCAP 1.0 to support the storage and playback of full resolution video. Implementations MAY use storage architectures for DVR content that differ from a general purpose file system. For this reason, a new volume type, `MediaStorageVolume`, is introduced for this storage.

6.5.4.1 Storage Management

The `MediaStorageOption` and `MediaStorageVolume` interfaces expose the special characteristics of DVR storage to applications. When a device is attached, the implementation SHALL determine whether it is supported for DVR media content storage. Implementations SHALL provide the `MediaStorageOption` via the `getOptions()` method on `StorageProxy` objects that are DVR media-capable.

The `MediaStorageOption` interface supports the creation of volumes for either DVR media storage or for general purpose file use. Volumes of the former type implement the `MediaStorageVolume` interface, which extends the `LogicalStorageVolume` interface and provides the ability to preallocate storage for a volume. Implementations SHALL support the creation of multiple instances of both `LogicalStorageVolume` and `MediaStorageVolume` on any `StorageProxy` that is capable of storing DVR content. Storage allocated for a `MediaStorageVolume` SHALL always be available for use by recordings created on that `MediaStorageVolume` until that storage is explicitly released by the specification of a smaller allocation for the volume.

The first media storage volume that is created on a `StorageProxy` is the default recording volume and is used to record programs for which a destination `MediaStorageVolume` is not specified for a recording. If there are multiple media-capable `StorageProxy` objects available with default recording volumes, the implementation may choose any default recording volume as the destination for such a recording. An implementation SHALL NOT spread the content for a recording across multiple storage volumes.

Implementations are not required to make files on DVR media volumes visible through `java.io`. Since media content is referenced through locators generated by the platform, an implementation MAY use multiple platform-specific files to represent a single media locator and MAY use storage architectures that differ significantly from typical file systems. Implementations SHALL support recordings of any length up to the available storage. The implementation SHALL ensure that the actual bandwidth available for DVR usage to `StorageProxies` always meets the combined maximum recording and playback capacity of the host device. The implementation SHALL delete any recordings which due to corruption it can determine cannot be played at all. The implementation SHALL delete any ancillary or index files related to a recording when a recording is lost or deleted.

6.5.4.2 Storage Initialization

Implementations that do not use a common storage architecture (e.g. file system type) for both general purpose files stored in `LogicalStorageVolumes` and media content stored in `MediaStorageVolumes` may not be able to dynamically shift storage between one use and the other without destroying content (e.g. repartitioning). An implementation supporting dynamic allocation is preferred, but an implementation MAY divide the storage on the device between the two uses in either a fixed or a dynamic fashion.

The `MediaStorageOption` provides an overloaded `initialize()` method that allows an highly privileged application to specify the amount of space to be allocated to each use. Implementations SHALL support the reallocation of space between the two uses. Because an OCAP application may change the allocation even on internal storage devices, implementations that cannot dynamically shift storage between the two uses SHOULD store any internal files on a separate reserved section of the device that would not be affected by a reallocation of storage. Implementations that cannot dynamically shift storage between the uses SHOULD NOT initialize a `StorageProxy` capable of DVR media storage until explicitly requested by an application. Implementations that do initialize such a `StorageProxy` before explicitly requested SHALL allocate for general purpose `LogicalStorageVolumes` at least 3% of the combined storage that can be allocated both uses. However, implementations are not required to preallocate more than 1GB to general purpose `LogicalStorageVolumes`.

6.6 Security

This section describes security features particular to the OCAP DVR platform. These are enhancements to the security features defined in OCAP 1.0 [2].

6.6.1 Permissions

The Java security model includes the means to restrict access to an API to only applications that have requested and been granted specific permission to access the API. The OCAP DVR API defines a small set of permissions to restrict access to certain APIs.

`MonitorAppPermission("recording")`

is required for an application attempting to manipulate a recording when the application is not the owner of implied recording.

6.6.2 Access scope of recordings

Applications MAY indicate an access scope at the time a recording is added to the recording database. An application MAY allow any other application to play back a recording it initiates, it MAY restrict playback to applications from a specific set of organizations, it MAY restrict playback to applications that are from the same organization, or it MAY restrict playback to itself. The implementation SHALL respect scope parameters, and not decrypt, decode, and display recordings that are not within the defined scope (see `org.ocap.storage.ExtendedFileAccessPermission` and the overloaded `org.ocap.dvr.RecordingManager.record()` method).

6.6.3 Content Protection

The OCAP DVR platform represents an extension of the cable network operator's service. As such, the platform respects the copy-protection rules defined in OCAP 1.0 [2]. Because recorded content originates from the cable network, all content SHALL be recorded in a fashion that ties the content to the network on which it was recorded. Any network-specific information saved by the Host device for this purpose must either be encrypted, or placed in a secure storage device location that cannot be read from outside the device. Furthermore, in order to meet copy-control requirements, all content SHALL be encrypted in a fashion that ties the content to the device which initiated its recording. These features are in general transparent to a viewer, as long as the recording device is not connected to a different network or a storage device is not connected to a different receiver.

Implementations SHALL NOT modify CCI bits associated with a recorded program.

The means by which a recording is associated with the network and the device from which it was recorded will be defined in a future revision of this specification.

6.6.4 Minimum Security Constraints

Minimum "level of security" requirements are asserted by this specification as follows:

- For DES implementations - The Triple DES Encryption Algorithm (TDEA) as per FIPS-46-3 [4] SHALL be the minimum DES configuration.
- For AES implementations - The minimum configuration SHALL be AES-192 as defined by FIPS-197 [6].

This specification complies with the OpenCable System Security Specification [1] regarding FIPS-140-2 [5] Level 1 security requirements and extends those requirements to any recording components.

6.7 OCAP DVR API

The OCAP DVR platform extends the OCAP-J API defined in OCAP 1.0 [2]. The additional packages, classes, and interfaces are listed here. For a complete definition and description of the APIs, see Annex A, *org.ocap.dvr*, Annex B, *org.ocap.dvr.navigation*, Annex C, *org.ocap.media*, and Annex D, *org.ocap.storage*. An implementation of the OCAP DVR platform SHALL include all of the packages, classes, and interfaces defined in OCAP 1.0 [2] as well as the following packages, classes, and interfaces.

6.7.1 Additions to OCAP 1.0 Packages

The OCAP DVR platform adds the following Java interfaces and classes to packages defined in OCAP 1.0 [2]:

`org.ocap.media.TimeShiftControl`

`org.ocap.storage.AllocateTimeShiftBufferOption`

org.ocap.storage.MediaStorageOption
org.ocap.storage.MediaStorageVolume
org.ocap.storage.SpaceAllocationHandler
org.ocap.storage.TimeShiftBufferOption

6.7.2 OCAP DVR Packages

The following packages are defined by the OCAP DVR platform:

org.ocap.dvr
org.ocap.dvr.navigation

7 Minimum Platform Capabilities

This chapter describes the minimum platform capabilities of the OCAP DVR platform.

7.1 Bit Rate

Implementations SHALL support three recording bit-rates; low, medium, and high. Where high causes a recording to be encoded with the best audio/video quality, but that takes more storage space than the other settings. For analog recordings these values are implementation specific. Bit-rates for digital recordings are related to transrating. Transrating is a process of modifying MPEG compression to achieve greater compression. Transrating is optional, but when supported, high bit-rate is equivalent to no transrating, and medium to low bit rates specify increasing compression. When transrating is supported, the type of transrating supported is implementation specific.

7.2 Playback Rates

Implementations SHALL support the following playback rates at a minimum: 16x, -8x, -4x, -2x, -1x, 0, 0.5x, 1x, 2x, 4x, 8x, 16x. For the -1x rate, it is allowed to only display i-frames.

7.3 OpenCable Set-top Terminal Core Requirements

The resources as specified for all OCAP profiles in the OC-SP-HOST-CFR-I16-040402 [3]) are REQUIRED. *This is a place holder for reference to DVR CFR.*

This page intentionally left blank.

Annex A org.ocap.dvr

This annex presents the org.ocap.dvr APIs.

Package org.ocap.dvr

Class Summary	
Interfaces	
RecordedService	This interface represents a service that was recorded for a period of time.
RecordingAlertListener	Listener for Recording Alerts.
RecordingListEntry	An Entry in the recording list database maintained by the RecordingManager.
RecordingListListener	Listener to receive changes in the recording list maintained by the RecordingManager.
RecordingManager	RecordingManager represents the entity that performs recordings.
RecordingResourceUsage	This interface represents a grouping of resources specific to an recording function performed by an application.
RecordingSession	This interface represents a recording session where a record has been scheduled or started by an application which called one of the RecordingManager record methods.
Classes	
RecordingAlertEvent	Event notifying that a scheduled Recording is about to occur.
RecordingListEvent	Event to notify changes in the list of recordings maintained by the RecordingManager.
RecordingTerminatedEvent	An Event Notifying that Recording has terminated for the ServiceContext.
Exceptions	
NoMoreDataEntriesException	No more data entries allowed for this recording list entry.
RecordingFailedException	This exception is returned when applications calls the getFailedException() for a failed recording session or an incomplete recording session.

org.ocap.dvr NoMoreDataEntriesException

Declaration

```
public class NoMoreDataEntriesException extends java.lang.Exception
```

```
java.lang.Object
|
+-- java.lang.Throwable
|
+-- java.lang.Exception
|
+-- org.ocap.dvr.NoMoreDataEntriesException
```

All Implemented Interfaces: java.io.Serializable

Description

No more data entries allowed for this recording list entry.

Member Summary

Constructors

```
NoMoreDataEntriesException()
Constructs a NoMoreDataEntriesException with no detail message
```

Constructors

NoMoreDataEntriesException()

```
public NoMoreDataEntriesException()
```

Constructs a NoMoreDataEntriesException with no detail message

org.ocap.dvr RecordedService

Declaration

```
public interface RecordedService extends javax.tv.service.Service
```

All Superinterfaces: javax.tv.service.Service

Description

This interface represents a service that was recorded for a period of time. As soon as recording begins a recorded service will be created. If the recording fails for any reason the recording shall be closed normally and the recorded service shall be available containing however much of the service was recorded.

Member Summary

Methods

RecordingListEntry	getRecordingListEntry()	Gets the RecordingListEntry corresponding to the RecordedService.
boolean	isDecodable()	Determines if the recording has a format which can be decoded for presentation by the implementation, e.g.
boolean	isDecryptable()	Determines if the recording can be decrypted by the implementation on the current network.

Methods

getRecordingListEntry()

```
public org.ocap.dvr.RecordingListEntry getRecordingListEntry()
```

Gets the RecordingListEntry corresponding to the RecordedService.

Returns: The RecordingListEntry for the service.

isDecodable()

```
public boolean isDecodable()
```

Determines if the recording has a format which can be decoded for presentation by the implementation, e.g. the bit rate, resolution, and encoding are supported.

Returns: True if the recording can be decoded, otherwise returns false.

isDecryptable()

```
public boolean isDecryptable()
```

Determines if the recording can be decrypted by the implementation on the current network.

Returns: True if the recording can be decrypted, otherwise returns false.

org.ocap.dvr RecordingAlertEvent

Declaration

```
public class RecordingAlertEvent extends java.util.EventObject
```

```
java.lang.Object
|
+-- java.util.EventObject
|
+-- org.ocap.dvr.RecordingAlertEvent
```

All Implemented Interfaces: java.io.Serializable

Description

Event notifying that a scheduled Recording is about to occur. This event is triggered for PendingRecordings and InConflictRecordings.

Member Summary	
Constructors	
	RecordingAlertEvent(RecordingListEntry source) Constructs the event.
Methods	
RecordingListEntry	getRecordingListEntry() Returns the RecordingListEntry that caused the event.

Constructors

RecordingAlertEvent(RecordingListEntry)

```
public RecordingAlertEvent(org.ocap.dvr.RecordingListEntry source)
```

Constructs the event.

Parameters:

source - The RecordingListEntry that caused the event.

Methods

getRecordingListEntry()

```
public org.ocap.dvr.RecordingListEntry getRecordingListEntry()
```

Returns the RecordingListEntry that caused the event.

Returns: The RecordingListEntry that caused the event.

org.ocap.dvr RecordingAlertListener

Declaration

```
public interface RecordingAlertListener extends java.util.EventListener
```

All Superinterfaces: java.util.EventListener

Description

Listener for Recording Alerts.

Member Summary

Methods

```
void recordingAlert(RecordingAlertEvent e)  
    Notifies the RecordingAlertListener that a scheduled activity is about to  
    happen.
```

Methods

recordingAlert(RecordingAlertEvent)

```
public void recordingAlert(org.ocap.dvr.RecordingAlertEvent e)
```

Notifies the RecordingAlertListener that a scheduled activity is about to happen.

Parameters:

e - The generated event.

org.ocap.dvr RecordingFailedException

Declaration

```
public class RecordingFailedException extends java.lang.Exception
```

```
java.lang.Object
|
+-- java.lang.Throwable
|
+-- java.lang.Exception
|
+-- org.ocap.dvr.RecordingFailedException
```

All Implemented Interfaces: java.io.Serializable

Description

This exception is returned when applications calls the getFailedException() for a failed recording session or an incomplete recording session.

Member Summary

Fields

static int	ACCESS_WITHDRAWN	Reason code : Recording did not complete successfully because access to the service or some component of it were withdrawn by the system before the scheduled completion of the recording.
static int	CA_REFUSAL	Reason code : Recording failed due to the CA system refusing to permit it.
static int	CONTENT_NOT_FOUND	Reason code : Recording failed because the requested content could not be found in the network
static int	INSUFFICIENT_RESOURCES	Reason code : Recording failed due to a lack of resources required to present this service.
static int	OUT_OF_BANDWIDTH	Reason code : Recording failed due lack of IO bandwidth to record this program.
static int	RESOURCES_REMOVED	Reason code : Recording did not complete successfully because Resources needed to present the service were removed before the scheduled completion of the recording.
static int	SERVICE_VANISHED	Reason code : Recording did not complete successfully because the service vanished from the network before the completion of the recording.
static int	SPACE_FULL	Reason code : Recording could not complete due to lack of storage space.
static int	TUNED_AWAY	Reason code : Recording did not complete successfully because the application selected another service on the service context.
static int	TUNING_FAILURE	Reason code : Recording failed due to problems with tuning.

Member Summary	
<pre>static int USER_STOP</pre>	Reason code : The application terminated the recording using RecordingSession.stop method or by calling the stop on the service context (if the recording was initiated using the RecordingManager.record(ServiceContext, ..)).
Methods	
<pre>int getReason()</pre>	Reports the reason for which the recording failed to complete successfully.

Fields

ACCESS_WITHDRAWN

```
public static final int ACCESS_WITHDRAWN
```

Reason code : Recording did not complete successfully because access to the service or some component of it were withdrawn by the system before the scheduled completion of the recording.

CA_REFUSAL

```
public static final int CA_REFUSAL
```

Reason code : Recording failed due to the CA system refusing to permit it.

CONTENT_NOT_FOUND

```
public static final int CONTENT_NOT_FOUND
```

Reason code : Recording failed because the requested content could not be found in the network

INSUFFICIENT_RESOURCES

```
public static final int INSUFFICIENT_RESOURCES
```

Reason code : Recording failed due to a lack of resources required to present this service.

OUT_OF_BANDWIDTH

```
public static final int OUT_OF_BANDWIDTH
```

Reason code : Recording failed due lack of IO bandwidth to record this program.

RESOURCES_REMOVED

```
public static final int RESOURCES_REMOVED
```

Reason code : Recording did not complete successfully because Resources needed to present the service were removed before the scheduled completion of the recording.

SERVICE_VANISHED

```
public static final int SERVICE_VANISHED
```

Reason code : Recording did not complete successfully because the service vanished from the network before the completion of the recording.

SPACE_FULL

```
public static final int SPACE_FULL
```

Reason code : Recording could not complete due to lack of storage space.

TUNED_AWAY

```
public static final int TUNED_AWAY
```

Reason code : Recording did not complete successfully because the application selected another service on the service context. This is applicable only if the recording was initiated using the RecordingManager.record(ServiceContext, ..)

TUNING_FAILURE

```
public static final int TUNING_FAILURE
```

Reason code : Recording failed due to problems with tuning.

USER_STOP

```
public static final int USER_STOP
```

Reason code : The application terminated the recording using RecordingSession.stop method or by calling the stop on the service context (if the recording was initiated using the RecordingManager.record(ServiceContext, ..)).

Methods

getReason()

```
public int getReason()
```

Reports the reason for which the recording failed to complete successfully.

Returns: the reason code for which the recording failed to complete successfully.

org.ocap.dvr RecordingListEntry

Declaration

```
public interface RecordingListEntry
```

Description

An Entry in the recording list database maintained by the RecordingManager. This entry may represent a pending recording, an ongoing recording, a completed recording, a failed recording, or a scheduled recording that would not be recorded due to conflict.

Member Summary

Methods

	<code>void addAppData(int dataID, byte[] data)</code>	Add application specific private data.
	<code>byte[] getAppData(int dataID)</code>	Get application data corresponding to specified dataID.
	<code>int[] getAppDataIDs()</code>	Get all Application specific data associated with this recording.
	<code>getAppID()</code>	Gets the application identifier of the application that owns this recording.
<code>org.dvb.application.AppID</code>		
<code>RecordingSession</code>	<code>getSession()</code>	Gets the RecordingSession associated with this entry.
	<code>void removeAppData(int dataID)</code>	Remove Application specific private data corresponding to the specified dataID.

Methods

addAppData(int, byte[])

```
public void addAppData(int dataID, byte[] data)
    throws NoMoreDataEntriesException
```

Add application specific private data. If the dataID is already in use, the data corresponding to dataID is overwritten.

Throws:

`java.lang.SecurityException` - if the caller does not have write permission for the recording.

`java.lang.IllegalArgumentException` - if the size of the byte array is more than the size supported by the implementation. The implementation shall support byte arrays of at least 256 bytes.

`NoMoreDataEntriesException` - if the recording list entry is unable to store any more Application data. The implementation shall support atleast 16 data entries per recording list entry.

getAppData(int)

```
public byte[] getAppData(int dataID)
```

Get application data corresponding to specified dataID.

Returns: the application data corresponding to the specified dataID; Null if there if no data corresponding to the specified dataID.

getAppDataIDs()

```
public int[] getAppDataIDs()
```

Get all Application specific data associated with this recording.

Returns: All application data corresponding to the specified dataID; Null if there if no application data.

getAppID()

```
public org.dvb.application.AppID getAppID()
```

Gets the application identifier of the application that owns this recording.

Returns: Application identifier of the owning application.

getSession()

```
public org.ocap.dvr.RecordingSession getSession()
```

Gets the RecordingSession associated with this entry.

Returns: The recording session associated with this entry.

removeAppData(int)

```
public void removeAppData(int dataID)
```

Remove Application specific private data corresponding to the specified dataID. This method exits silently if there was no data corresponding to the dataID.

Throws:

`java.lang.SecurityException` - is the application does not have write permission for the recording.

org.ocap.dvr RecordingListEvent

Declaration

```
public class RecordingListEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.ocap.dvr.RecordingListEvent
```

All Implemented Interfaces: java.io.Serializable

Description

Event to notify changes in the list of recordings maintained by the RecordingManager. When a recording starts there would be two events - an ENTRY_DELETED for PendingRecording or InConflictRecording and an ENTRY_ADDED for an InProgressRecording.

Member Summary	
Fields	
static int	ENTRY_ADDED A new RecordingListEntry was added.
static int	ENTRY_DELETED A RecordingListEntry was deleted.
static int	ENTRY_STATE_CHANGED The state of a RecordingListEntry changed
Constructors	
	RecordingListEvent(RecordingListEntry source, int newState, int oldState) Constructs the event.
Methods	
int	getChange() Returns the change to the RecordingListEntry.
int	getOldState() Returns the old state for the RecordingListEntry.
RecordingListEntry	getRecordingListEntry() Returns the RecordingListEntry that caused the event.
int	getState() Returns the new state for the RecordingListEntry.

Fields

ENTRY_ADDED

```
public static final int ENTRY_ADDED
```

A new `RecordingListEntry` was added.

ENTRY_DELETED

```
public static final int ENTRY_DELETED
```

A `RecordingListEntry` was deleted.

ENTRY_STATE_CHANGED

```
public static final int ENTRY_STATE_CHANGED
```

The state of a `RecordingListEntry` changed

Constructors

RecordingListEvent(RecordingListEntry, int, int)

```
public RecordingListEvent(org.ocap.dvr.RecordingListEntry source, int newState,  
                          int oldState)
```

Constructs the event.

Parameters:

`source` - The `RecordingListEntry` that caused the event.

Methods

getChange()

```
public int getChange()
```

Returns the change to the `RecordingListEntry`.

Returns: whether the entry was added, deleted or modified.

getOldState()

```
public int getOldState()
```

Returns the old state for the `RecordingListEntry`.

Returns: The old state.

getRecordingListEntry()

```
public org.ocap.dvr.RecordingListEntry getRecordingListEntry()
```

Returns the `RecordingListEntry` that caused the event.

Returns: The `RecordingListEntry` that caused the event.

getState()

```
public int getState()
```

Returns the new state for the `RecordingListEntry`.

Returns: The new state.

org.ocap.dvr RecordingListListener

Declaration

```
public interface RecordingListListener extends java.util.EventListener
```

All Superinterfaces: java.util.EventListener

Description

Listener to receive changes in the recording list maintained by the RecordingManager.

Member Summary

Methods

```
void recordingListUpdate(RecordingListEvent e)  
    Notifies the RecordingListListener of an event generated by the  
    RecordingManager.
```

Methods

recordingListUpdate(RecordingListEvent)

```
public void recordingListUpdate(org.ocap.dvr.RecordingListEvent e)
```

Notifies the RecordingListListener of an event generated by the RecordingManager. Events are generated when there are changes in the RecordingList.

Parameters:

e - The generated event.

org.ocap.dvr RecordingManager

Declaration

```
public interface RecordingManager
```

Description

RecordingManager represents the entity that performs recordings.

Member Summary

Fields

```
static byte HIGH_BIT_RATE
    Indicates an implementation specific value for high resolution or bit-rate.
static byte LOW_BIT_RATE
    Indicates an implementation specific value for low resolution or bit-rate.
static byte MEDIUM_BIT_RATE
    Indicates an implementation specific value for medium resolution or bit-rate.
static byte RECORD_IF_NO_CONFLICTS
    Record only if there are no conflicts.
static byte RECORD_WITH_CONFLICTS
    Record when resource conflicts exist.
```

Methods

```
void addRecordingAlertListener(RecordingAlertListener ral)
    Adds an event listener for receiving events corresponding to a transition from a
    pending state to an in-progress state or a failed state.
void addRecordingAlertListener(RecordingAlertListener ral, long
    alertBefore)
    Adds an event listener for receiving events corresponding to a transition from a
    pending state to an in-progress state or a failed state.
void addRecordingListListener(RecordingListListener rll)
    Adds an event listener for changes in status of recording list entries.
void delete(RecordingListEntry rle)
    Deletes the recording from storage.
getEntries()
    Gets the list of entries maintained by the RecordingManager.
org.ocap.dvr.navigation.RecordingList
    getEntries(org.ocap.dvr.navigation.RecordingListFilter
    filter)
    Gets the list of recording matching the specified filter.
org.ocap.dvr.navigation.RecordingList
    RecordingListEntry
    record(org.ocap.net.OcapLocator source, long startTime, long
    duration, long resolution, java.util.Date expirationDate,
    byte priorityFlag,
    org.ocap.storage.ExtendedFileAccessPermissions access,
    java.lang.String organization,
    org.ocap.storage.MediaStorageVolume destination)
    Encrypts and records the stream or streams indicated by the source
    org.ocap.net.OcapLocator array parameter.
```

Member Summary

RecordingListEntry	<pre>record(javax.tv.service.selection.ServiceContext serviceContext, long starTime, long duration, long resolution, java.util.Date expirationDate, byte priorityFlag, org.ocap.storage.ExtendedFileAccessPermissions access, java.lang.String organization, org.ocap.storage.MediaStorageVolume destination) Encrypts and records the stream or streams that are being presented in the ServiceContext indicated by the serviceContext parameter.</pre>
void	<pre>removeRecordingAlertListener(RecordingAlertListener ral) Removes a registered event listener for receiving recording events.</pre>
void	<pre>removeRecordingListListener(RecordingListListener rll) Removes a registered event listener for changes in status of recording list entries.</pre>
void	<pre>setSpaceAllocationHandler(org.ocap.storage.SpaceAllocationHan dler sah) Set the SpaceAllocationHandler that will be invoked when any application attempts to allocate space in any MediaStorageVolume.</pre>

Fields

HIGH_BIT_RATE

```
public static final byte HIGH_BIT_RATE
```

Indicates an implementation specific value for high resolution or bit-rate.

LOW_BIT_RATE

```
public static final byte LOW_BIT_RATE
```

Indicates an implementation specific value for low resolution or bit-rate.

MEDIUM_BIT_RATE

```
public static final byte MEDIUM_BIT_RATE
```

Indicates an implementation specific value for medium resolution or bit-rate.

RECORD_IF_NO_CONFLICTS

```
public static final byte RECORD_IF_NO_CONFLICTS
```

Record only if there are no conflicts. When used as the `priorityFlag` to the `record` method, the recording is scheduled only if there are no conflicts. If there are conflicts the `record` method will return a `RecordingListEntry` with status set to `STATUS_SCHEDULE_CONFLICT`.

RECORD_WITH_CONFLICTS

```
public static final byte RECORD_WITH_CONFLICTS
```

Record when resource conflicts exist. This value could be used as the `priorityFlag` parameter value to the `record` methods. When this is used as a priority value, if the request conflicts with another `RecordingListEntry`, resources are resolved using recording resource conflict resolution rules.

Methods

addRecordingAlertListener(RecordingAlertListener)

```
public void addRecordingAlertListener(org.ocap.dvr.RecordingAlertListener ral)
```

Adds an event listener for receiving events corresponding to a transition from a pending state to an in-progress state or a failed state. The listener parameter will only be informed of these events for entries the calling application has read file access permission to, or all of these events if the calling application has MonitorAppPermission("recording").

Parameters:

`ral` - The listener to be registered.

addRecordingAlertListener(RecordingAlertListener, long)

```
public void addRecordingAlertListener(org.ocap.dvr.RecordingAlertListener ral,
                                     long alertBefore)
```

Adds an event listener for receiving events corresponding to a transition from a pending state to an in-progress state or a failed state. The listener parameter will only be informed of these events for entries the calling application has read file access permission to, or all of these events if the calling application has MonitorAppPermission("recording").

Parameters:

`ral` - The listener to be registered.

`alertBefore` - Time in milliseconds for the alert to be generated before the start of the scheduled event.

addRecordingListListener(RecordingListListener)

```
public void addRecordingListListener(org.ocap.dvr.RecordingListListener rll)
```

Adds an event listener for changes in status of recording list entries. The listener parameter will only be informed of changes that affect entries the calling application has read file access permission to, or all events if the calling application has MonitorAppPermission("recording").

Parameters:

`rll` - The listener to be registered.

delete(RecordingListEntry)

```
public void delete(org.ocap.dvr.RecordingListEntry rle)
```

Deletes the recording from storage. The method removes the `RecordingListEntry`, as well as the corresponding `RecordingSession`, `RecordedService` objects, and all recorded elementary streams (e.g., files and directory entries) associated with the `RecordedService`. If any application call any method on stale references of removed objects the implementation shall throw an `IllegalStateException`.

If the recording is in the `IN_PROGRESS` state the implementation will call the `RecordingSession` `stop` method before deleting the recording.

Parameters:

`rle` - The recording list entry to be deleted.

Throws:

`java.lang.SecurityException` - if the calling application does not have write file access permission, or `MonitorAppPermission("recording")`.

getEntries()

```
public org.ocap.dvr.navigation.RecordingList getEntries()
```

Gets the list of entries maintained by the `RecordingManager`. This list includes scheduled recordings, ongoing recordings, completed recordings and failed recordings.

Returns: an instance of `RecordingList` containing all the entries maintained by the `RecordingManager` to which the calling application has read file access permission. If the calling application has `MonitorAppPermission("recording")` all entries are returned.

getEntries(RecordingListFilter)

```
public org.ocap.dvr.navigation.RecordingList
    getEntries(org.ocap.dvr.navigation.RecordingListFilter filter)
```

Gets the list of recording matching the specified filter.

Returns: an instance of `RecordingList` containing all the entries matching the specified filter to which the calling application has read file access permission. If the calling application has `MonitorAppPermission("recording")` all matching entries are returned.

record(OcapLocator[], long, long, long, Date, byte, ExtendedFileAccessPermissions, String, MediaStorageVolume)

```
public org.ocap.dvr.RecordingListEntry record(org.ocap.net.OcapLocator[] source,
    long startTime, long duration, long resolution,
    java.util.Date expirationDate, byte priorityFlag,
    org.ocap.storage.ExtendedFileAccessPermissions access,
    java.lang.String organization,
    org.ocap.storage.MediaStorageVolume destination)
    throws IllegalArgumentException, InvalidServiceComponentException
```

Encrypts and records the stream or streams indicated by the source

`org.ocap.net.OcapLocator` array parameter. If the source parameter contains a locator without an elementary stream term, all audio, video, closed captioning, and content advisory elementary streams implied by the locator are recorded. Recording of other streams is optional, but not necessary for playback. If multiple locators are contained within the source parameter, all of them MUST be part of the same service.

Parameters:

`source` - Source of streams to be recorded.

`startTime` - Start time of the recording. Start time for this method should be either the current time or a time in the future.

`duration` - Length of time to record in milli-seconds.

`resolution` - An application may specify `LOW_BIT_RATE`, `MEDIUM_BIT_RATE`, or `HIGH_BIT_RATE`. For analog recordings the corresponding bit-rate values are implementation specific. For digital recordings these values request optional transrating. When transrating is supported, `HIGH_BIT_RATE` indicates no transrating, and `MEDIUM_BIT_RATE` to `LOW_BIT_RATE` indicates increasing compression with a potential decrease in video quality.

`expirationDate` - Date the recording becomes invalid.

`priorityFlag` - Indication whether the recording should be made regardless of resource conflict or not. This parameter can contain the values `RECORD_IF_NO_CONFLICTS`, or `RECORD_WITH_CONFLICTS`.

`access` - File access permission for the recording.

`organization` - Name of the organization this recording will be tied to. Used to authenticate playback applications by matching this parameter to an organization name field in any playback application's certificate chain. Can be set to null to disable this playback application authentication.

`destination` - The volume that represents the Storage location of the recording. If the value of this parameter is null, the implementation SHALL use the default recording volume (see `org.ocap.storage.MediaStorageOption`) in one of the storage devices connected.

Throws:

`java.lang.SecurityException` - if the application does not have `MonitorAppPermission("record")`, or if the application does not have permission to access the service context.

`java.lang.IllegalArgumentException` - if the `startTime` is in the past, or if duration is negative, or if resolution does not equal one of `LOW_BIT_RATE`, `MEDIUM_BIT_RATE`, or `HIGH_BIT_RATE`, or if `priorityFlag` does not contain the value `RECORD_IF_NO_CONFLICTS` or `RECORD_WITH_CONFLICTS`, or if `organization` is not found in the application's certificate file.

`javax.tv.service.selection.InvalidServiceComponentException` - if all of the locators in the source parameter are not all in the same service.

record(ServiceContext, long, long, long, Date, byte, ExtendedFileAccessPermissions, String, MediaStorageVolume)

```
public org.ocap.dvr.RecordingListEntry
    record(javax.tv.service.selection.ServiceContext serviceContext,
           long startTime, long duration, long resolution,
           java.util.Date expirationDate, byte priorityFlag,
           org.ocap.storage.ExtendedFileAccessPermissions access,
           java.lang.String organization,
           org.ocap.storage.MediaStorageVolume destination)
    throws IllegalArgumentException
```

Encrypts and records the stream or streams that are being presented in the `ServiceContext` indicated by the `serviceContext` parameter. If the Service being recorded from is tuned away from, the recording SHALL be terminated.

If the `startTime` is in the past and the source

`javax.tv.service.selection.ServiceContext` can be associated with a `org.ocap.storage.TimeShiftBufferOption`, the contents of the time-shift buffer are immediately stored to the destination beginning at the `startTime`, if possible, up to the live broadcast point. If the time-shift buffer does not contain the source from the `startTime`, as much of the source is recorded as possible. If the `startTime` is in the past, but a time-shift buffer cannot be associated with the recording, the recording begins from the live broadcast point. From there the contents of the live broadcast are recorded until the remaining duration is satisfied.

Parameters:

`serviceContext` - The ServiceContext to record from.

`startTime` - Start time of the recording. Start time for this method should either be the current time or any time in the past.

`duration` - Length of time to record in milli-seconds.

`resolution` - An application may specify `LOW_BIT_RATE`, `MEDIUM_BIT_RATE`, or `HIGH_BIT_RATE`. For analog recordings the corresponding bit-rate values are implementation specific. For digital recordings these values request optional transrating. When transrating is supported, `HIGH_BIT_RATE` indicates no transrating, and `MEDIUM_BIT_RATE` to `LOW_BIT_RATE` indicates increasing compression with a potential decrease in video quality.

`expirationDate` - Date the recording becomes invalid.

`priorityFlag` - Indication whether the recording should be made regardless of resource conflict or not. This parameter can contain the values `RECORD_IF_NO_CONFLICTS`, or `RECORD_WITH_CONFLICTS`.

`access` - File access permission for the recording.

`organization` - Name of the organization this recording will be tied to. Used to authenticate playback applications by matching this parameter to an organization name field in any playback application's certificate chain. Can be set to null to disable this playback application authentication.

`destination` - The volume that represents the Storage location of the recording. If the `MediaVolumeStorage` is null the storage associated with any time-shift buffer associated with the `ServiceContext` will be used, or if no time-shift buffer is associated with the `ServiceContext`, a storage volume configured as the default SHALL be used provided it is configured for AVFS, in which case the implementation shall pick the storage device. If `MediaVolumeStorage` is not null and the `ServiceContext` is associated with a time-shift buffer and the time-shift buffer is not attached to the `MediaVolumeStorage` then an `IllegalArgumentException` SHALL be thrown.

Throws:

`java.lang.SecurityException` - if the application does not have `MonitorAppPermission("record")`, or if the application does not have permission to access the service context.

`java.lang.IllegalArgumentException` - if `serviceContext` is not presenting a broadcast service, or `duration` is negative, or if `resolution` does not equal one of `LOW_BIT_RATE`, `MEDIUM_BIT_RATE`, or `HIGH_BIT_RATE`, or if `priorityFlag` does not contain the value `RECORD_IF_NO_CONFLICTS` or `RECORD_WITH_CONFLICTS`, or if `organization` is not found in the application's certificate file, or if the start time is in the future.

removeRecordingAlertListener(RecordingAlertListener)

```
public void removeRecordingAlertListener(org.ocap.dvr.RecordingAlertListener ral)
```

Removes a registered event listener for receiving recording events. If the listener specified is not registered then this method has no effect.

Parameters:

`ral` - the listener to be removed.

removeRecordingListListener(RecordingListListener)

```
public void removeRecordingListListener(org.ocap.dvr.RecordingListListener rll)
```

Removes a registered event listener for changes in status of recording list entries. If the listener specified is not registered then this method has no effect.

Parameters:

rll - the listener to be removed.

setSpaceAllocationHandler(SpaceAllocationHandler)

```
public void setSpaceAllocationHandler(org.ocap.storage.SpaceAllocationHandler sah)
```

Set the SpaceAllocationHandler that will be invoked when any application attempts to allocate space in any MediaStorageVolume. At most only one instance of this handler can be set. Subsequent calls to this method replaces the previous instance with the new one.

Parameters:

sah - the space reservation handler.

Throws:

java.lang.SecurityException - if the caller does not have MonitorAppPermission("recording").

org.ocap.dvr RecordingResourceUsage

Declaration

```
public interface RecordingResourceUsage
```

Description

This interface represents a grouping of resources specific to an recording function performed by an application.

Member Summary

Methods

```
RecordingListEntry getRecordingListEntry()  
    Gets the RecordingListEntry associated with the set of resources contained in  
    the usage and initiated by the application returned by the base  
    ResourceUsage.getAppID method.
```

Methods

getRecordingListEntry()

```
public org.ocap.dvr.RecordingListEntry getRecordingListEntry()
```

Gets the RecordingListEntry associated with the set of resources contained in the usage and initiated by the application returned by the base ResourceUsage.getAppID method.

Returns: The recording list entry associated with the usage and application.

org.ocap.dvr RecordingSession

Declaration

```
public interface RecordingSession
```

Description

This interface represents a recording session where a record has been scheduled or started by an application which called one of the `RecordingManager` record methods. A recording session may be pending (i.e. waiting for the start-time to occur), in-progress, completed, or failed. While pending a session may be in conflict for resources with other sessions. Any such conflicts must be resolved before the scheduled start time of the session, if not, the pending session is expected to result in a failed session. When the implementation detects a schedule conflict, the implementation either resolves the conflict using the Application priority of the conflicting sessions, or invokes the `org.ocap.resource.ResourceContentionHandler` if one is set. The resolution of the conflict by the implementation or the `ResourceContentionHandler` will result in some of the overlapping sessions to be pending without conflict and some to be pending with conflict.

Member Summary

Fields

```
static int COMPLETED_STATE
    The Recording has completed successfully.

static int FAILED_STATE
    The Recording has failed.

static int IN_PROGRESS_INSUFFICIENT_SPACE_STATE
    The Recording has been initiated and is ongoing, but the implementation has detected
    that storage space may not be sufficient to complete the recording.

static int IN_PROGRESS_STATE
    The Recording has been initiated and is ongoing.

static int INCOMPLETE_STATE
    The Recording was initiated but failed in the IN_PROGRESS_STATE before the
    COMPLETED_STATE could be reached.

static int PENDING_NO_CONFLICT_STATE
    The Recording is Pending.

static int PENDING_WITH_CONFLICT_STATE
    The Recording may not go through due to conflicts.
```

Methods

```
void cancel()
    Cancels the recording session if the state is PENDING_WITH_CONFLICTS_STATE,
    or PENDING_NO_CONFLICTS_STATE.

getDestination()
    Gets the destination of the RecordedService contained within the
    org.ocap.storage.Media StorageVolume

java.util.Date getExpirationDate()
    Gets the expiration Date of the recording.
```

Member Summary

java.lang.Exception	getFailedException()	Gets the exception that caused the session to enter the FAILED_STATE, or INCOMPLETE_STATE.
	getFileAccessPermissions()	
org.ocap.storage.ExtendedFileAccessPermissions		Gets the file access permissions passed to the RecordingManager record method for the recording.
	getOverlappingEntries()	
org.ocap.dvr.navigation.RecordingList		Gets any other RecordingSession that overlaps with the duration of this RecordingSession.
byte	getPriorityFlag()	Returns the priority flag for the recording session.
long	getRecordedBitRate()	Get the bit-rate used for encoding and storage of this recorded service.
long	getRecordedDuration()	Gets the duration corresponding to the actual duration recorded.
long	getRecordedSize()	Gets the size of the recording in bytes.
byte	getRequestedBitRate()	Gets the bit-rate requested for encoding and storage of this recorded service as originally set by a call to one of the RecordingManager record methods.
long	getRequestedDuration()	Gets the duration requested for the recording.
RecordedService	getService()	Returns the RecordedService corresponding to the session.
long	getSpaceRequired()	Gets the space required for the recording by multiply the bytes per second requirement of the media type by the duration of the recording in seconds.
long	getStartTime()	Returns the start time corresponding to the start time parameter passed to a RecordingManager record method.
int	getState()	Returns the state of the RecordingSession.
void	setDuration(long duration)	Sets the duration of the RecordingListEntry.
void	setExpirationDate(java.util.Date date)	Sets the life span of the recording.
void	setFileAccessPermissions(org.ocap.storage.ExtendedFileAccessPermissions fap)	Sets the file access permissions of the recording.
void	setPriorityFlag(byte priority)	Changes the priority flag for the recording session.
void	setRequestedBitRate(byte bitRate)	Sets the bit-rate requested for encoding and storage of this recorded service.
void	stop()	Stops the recording normally, regardless of how much of the duration has been recorded.

Fields

COMPLETED_STATE

```
public static final int COMPLETED_STATE
```

The Recording has completed successfully.

FAILED_STATE

```
public static final int FAILED_STATE
```

The Recording has failed.

IN_PROGRESS_INSUFFICIENT_SPACE_STATE

```
public static final int IN_PROGRESS_INSUFFICIENT_SPACE_STATE
```

The Recording has been initiated and is ongoing, but the implementation has detected that storage space may not be sufficient to complete the recording. This situation may arise when the implementation detects that the storage space may not be sufficient to complete this recording session and other recording sessions that are in_progress. The recording is not expected to complete successfully. A recording session may enter this state from IN_PROGRESS_STATE or PENDING_NO_CONFLICT_STATE. Implementation may start a recording in IN_PROGRESS_STATE based on initial estimation for space required and later change to IN_PROGRESS_INSUFFICIENT_SPACE_STATE when the implementation has enough information to compute a more accurate estimation of space needed. If an implementation cannot detect insufficient space in advance, the implementation may start the recording in IN_PROGRESS_STATE and then transition to FAILED_STATE when space runs out. It is also possible for a recording to start in IN_PROGRESS_INSUFFICIENT_SPACE_STATE and later move to IN_PROGRESS_STATE or COMPLETED_STATE. This could occur if other recordings were deleted after the start of the recording or if implementation computes a better estimate of the space needed as the recording progresses.

IN_PROGRESS_STATE

```
public static final int IN_PROGRESS_STATE
```

The Recording has been initiated and is ongoing. The recording is expected to complete successfully.

INCOMPLETE_STATE

```
public static final int INCOMPLETE_STATE
```

The Recording was initiated but failed in the IN_PROGRESS_STATE before the COMPLETED_STATE could be reached. The RecordingSession will contain a RecordedService that can be played for whatever duration was recorded.

PENDING_NO_CONFLICT_STATE

```
public static final int PENDING_NO_CONFLICT_STATE
```

The Recording is Pending. The recording is expected to complete successfully.

PENDING_WITH_CONFLICT_STATE

```
public static final int PENDING_WITH_CONFLICT_STATE
```

The Recording may not implement due to conflicts. This could be due to a schedule conflict ResourceConflictException or space conflict SpaceFullException

Methods

cancel()

```
public void cancel()
    throws IllegalStateException
```

Cancels the recording session if the state is `PENDING_WITH_CONFLICTS_STATE`, or `PENDING_NO_CONFLICTS_STATE`. Once the recording is in progress, the stop method can be called. Success completion of this method causes the recording's `RecordingListEntry` to be deleted from the database. Cancelling a recording session may resolve one or more conflicts. In this case some pending recording sessions with conflicts may be changed to pending without conflicts.

Throws:

`java.lang.SecurityException` - if the calling application does not have write permission, or does not have `MonitorAppPermission("recording")`.

`java.lang.IllegalStateException` - if the state of the recording session is not `IN_PROGRESS_STATE`.

getDestination()

```
public org.ocap.storage.MediaStorageVolume getDestination()
    throws IllegalStateException
```

Gets the destination of the `RecordedService` contained within the `RecordingSession`.

Returns: Destination of the `RecordedService`. If no `RecordedService` was created and the `FAILURE_STATE` entered, this method returns null. If no destination was specified and the recording has not yet started, this method returns null.

Throws:

`java.lang.IllegalStateException` - if the recording session is not in the `IN_PROGRESS_STATE`, `COMPLETE_STATE`, or `FAILURE_STATE`.

getExpirationDate()

```
public java.util.Date getExpirationDate()
```

Gets the expiration Date of the recording.

Returns: Expiration Date of the recording.

getFailedException()

```
public java.lang.Exception getFailedException()
    throws IllegalStateException
```

Gets the exception that caused the session to enter the `FAILED_STATE`, or `INCOMPLETE_STATE`.

Returns: The exception that caused the failure. The exception returned will be a `RecordingFailedException`.

Throws:

`java.lang.IllegalStateException` - if the recording session is not in the `FAILED_STATE` or `INCOMPLETE_STATE`.

getFileAccessPermissions()

```
public org.ocap.storage.ExtendedFileAccessPermissions getFileAccessPermissions()
```

Gets the file access permissions passed to the `RecordingManager` `record` method for the recording.

Returns: File access permissions of the recording.

getOverlappingEntries()

```
public org.ocap.dvr.navigation.RecordingList getOverlappingEntries()  
    throws IllegalStateException
```

Gets any other `RecordingSession` that overlaps with the duration of this `RecordingSession`. This method will return null unless the session is in the `PENDING_WITH_CONFLICTS_STATE`, `PENDING_NO_CONFLICTS_STATE`, or `IN_PROGRESS_STATE`. The returned list will contain only `OverlappingRecordingSessions` for which the application has read access permission. The `RecordingList` returned is only a copy of the list of overlapping entries at the time of this method call. This list is not updated if there are any changes. A new call to this method will be required to get the updated list.

Throws:

`java.lang.IllegalStateException` - if the recording session is not in the `PENDING_WITH_CONFLICT_STATE`, `PENDING_NO_CONFLICTS_STATE`, or `IN_PROGRESS_STATE`.

getPriorityFlag()

```
public byte getPriorityFlag()
```

Returns the priority flag for the recording session.

Returns: priority flag of the recording session.

getRecordedBitRate()

```
public long getRecordedBitRate()
```

Get the bit-rate used for encoding and storage of this recorded service.

Returns: Bit-rate in bytes per second.

getRecordedDuration()

```
public long getRecordedDuration()
```

Gets the duration corresponding to the actual duration recorded.

Returns: The duration of the recording in milli-seconds.

getRecordedSize()

```
public long getRecordedSize()
```

Gets the size of the recording in bytes. In the `PENDING_WITH_CONFLICTS_STATE`, or `PENDING_NO_CONFLICTS_STATE` the value returned is always zero. In the `IN_PROGRESS_STATE` the value returned is an approximation as it will be changing as this method returns. In the `FAILED_STATE` the value will be zero or the size of the recording when the failure occurred. In the `COMPLETED_STATE` the size reflects the size of the entire intended recording.

getRequestedBitRate()

```
public byte getRequestedBitRate()
```

Gets the bit-rate requested for encoding and storage of this recorded service as originally set by a call to one of the `RecordingManager` record methods. Possible return values include `RecordingManager.LOW_BIT_RATE`, `RecordingManager.MEDIUM_BIT_RATE`, or `RecordingManager.HIGH_BIT_RATE`.

Returns: Bit-rate requested in bytes per second.

getRequestedDuration()

```
public long getRequestedDuration()
```

Gets the duration requested for the recording.

Returns: The duration of the recording in milli-seconds.

getService()

```
public org.ocap.dvr.RecordedService getService()  
    throws IllegalStateException
```

Returns the `RecordedService` corresponding to the session. If the recording is in the `PENDING_WITH_CONFLICTS_STATE`, or `PENDING_NO_CONFLICTS_STATE` and the `RecordedService` has not been created yet, this method returns null.

Returns: The recorded service associated with the recording session.

Throws:

```
java.lang.IllegalStateException
```

getSpaceRequired()

```
public long getSpaceRequired()
```

Gets the space required for the recording by multiplying the bytes per second requirement of the media type by the duration of the recording in seconds. Due to implementation specific handling, there is not guarantee the value returned from this method will match the value returned from `getSpaceOccupied` even when the recording session is in the `COMPLETE_STATE`.

Returns: Space required for the recording in bytes.

getStartTime()

```
public long getStartTime()
```

Returns the start time corresponding to the start time parameter passed to a `RecordingManager` record method.

Returns: Start time of the recording.

getState()

```
public int getState()
```

Returns the state of the `RecordingSession`.

Returns: State of the recording session.

setDuration(long)

```
public void setDuration(long duration)
    throws IllegalStateException
```

Sets the duration of the RecordingListEntry. This method is only effective when the session is in the PENDING_WITH_CONFLICTS_STATE, PENDING_NO_CONFLICTS_STATE, or IN_PROGRESS_STATE. If the change in duration generates a conflict, either this recording session or some other overlapping recording session may be changed to PENDING_WITH_CONFLICT state according to the conflict resolution rules.

Parameters:

duration - Length of time to record in milli-seconds.

Throws:

java.lang.SecurityException - if the calling application does not have write permission, or does not have MonitorAppPermission("recording").

java.lang.IllegalStateException - if the recording session is not in the PENDING_WITH_CONFLICTS_STATE, PENDING_NO_CONFLICTS_STATE, IN_PROGRESS_STATE or IN_PROGRESS_INSUFFICIENT_SPACE_STATE.

setExpirationDate(Date)

```
public void setExpirationDate(java.util.Date date)
```

Sets the life span of the recording. This method may change the life span set by the RecordingManager record method.

Parameters:

date - New expiration date.

Throws:

java.lang.SecurityException - if the caller is not the owner of the recording or does not have MonAppPermission("recording").

setFileAccessPermissions(ExtendedFileAccessPermissions)

```
public void setFileAccessPermissions(org.ocap.storage.ExtendedFileAccessPermissions
    fap)
```

Sets the file access permissions of the recording. This method may change the permissions set by the RecordingManager record method.

Parameters:

fap - New file access permissions.

Throws:

java.lang.SecurityException - if the caller is not the owner of the recording or does not have MonAppPermission("recording").

setPriorityFlag(byte)

```
public void setPriorityFlag(byte priority)
```

Changes the priority flag for the recording session. If the new value of the priority flag is RECORD_WITH_CONFLICTS and if the recording session is in PENDING_WITH_CONFLICTS_STATE, the conflict resolution rules will be applied to see if the conflict can be resolved in favor of this recording session. If the new value of the priority flag is

RECORD_IF_NO_CONFLICTS and if the recording session is in PENDING_NO_CONFLICT_STATE, this method may cause the recording session to be changed to PENDING_WITH_CONFLICTS_STATE.

Parameters:

priority - flag of the recording session.

setRequestedBitRate(byte)

```
public void setRequestedBitRate(byte bitRate)
    throws IllegalArgumentException, IllegalStateException
```

Sets the bit-rate requested for encoding and storage of this recorded service. The implementation MAY round the requested number down to the nearest supported value.

Parameters:

bitRate - Possible values include RecordingManager.LOW_BIT_RATE, RecordingManager.MEDIUM_BIT_RATE, or RecordingManager.HIGH_BIT_RATE.

Throws:

java.lang.IllegalArgumentException - if the bitRate parameter is not one of RecordingManager.LOW_BIT_RATE, RecordingManager.MEDIUM_BIT_RATE, or RecordingManager.HIGH_BIT_RATE.

java.lang.IllegalStateException - if the session is not in either the PENDING_WITH_CONFLICT_STATE or PENDING_NO_CONFLICTS_STATE.

stop()

```
public void stop()
    throws IllegalStateException
```

Stops the recording normally, regardless of how much of the duration has been recorded. Moves the session to the INCOMPLETE_STATE. Does nothing if the session is in the PENDING_WITH_CONFLICTS_STATE, PENDING_NO_CONFLICTS_STATE, or COMPLETED_STATE.

Throws:

java.lang.SecurityException - if the calling application does not have write permission, or does not have MonitorAppPermission("recording").

java.lang.IllegalStateException - if the recording session is not in the IN_PROGRESS_STATE, or IN_PROGRESS_INSUFFICIENT_SPACE_STATE.

org.ocap.dvr RecordingTerminatedEvent

Declaration

```
public class RecordingTerminatedEvent extends
    javax.tv.service.selection.ServiceContextEvent

java.lang.Object
|
+-- java.util.EventObject
    |
    +-- javax.tv.service.selection.ServiceContextEvent
        |
        +-- org.ocap.dvr.RecordingTerminatedEvent
```

All Implemented Interfaces: java.io.Serializable

Description

An Event Notifying that Recording has terminated for the ServiceContext. This event is generated by a ServiceContext that is presenting a time-shifted service or a service that is being recorded. The presentation is not yet terminated as the playback point is time-delayed. This event is generated only when the playback point is not the same as the live point. A PresentationTerminatedEvent will be generated when the playback point catches up with the point of record termination.

Member Summary

Fields

```
static int ACCESS_WITHDRAWN
    Reason code: Access to the service or some component of it has been withdrawn by
    the system.

static int RESOURCES_REMOVED
    Reason code: Resources needed to present the service have been removed.

static int SCHEDULED_STOP
    Reason code: The recording was terminated normally as scheduled.

static int SERVICE_VANISHED
    Reason code: The service vanished from the network.

static int USER_STOP
    Reason code: The user requested that the recording be stopped.
```

Constructors

```
RecordingTerminatedEvent( javax.tv.service.selection.ServiceCo
ntext source)
    Constructs the event.
```

Methods

```
int getReason()
    Returns the reason for the record termination.
```

Fields

ACCESS_WITHDRAWN

```
public static final int ACCESS_WITHDRAWN
```

Reason code: Access to the service or some component of it has been withdrawn by the system. An example of this is the end of a free preview period for IPPV content.

RESOURCES_REMOVED

```
public static final int RESOURCES_REMOVED
```

Reason code: Resources needed to present the service have been removed. Will be generated if the `javax.tv.service.selection.ServiceContext stop` method is called.

SCHEDULED_STOP

```
public static final int SCHEDULED_STOP
```

Reason code: The recording was terminated normally as scheduled.

SERVICE_VANISHED

```
public static final int SERVICE_VANISHED
```

Reason code: The service vanished from the network.

USER_STOP

```
public static final int USER_STOP
```

Reason code: The user requested that the recording be stopped. Also, if the `RecordingSession stop` method is called.

Constructors

RecordingTerminatedEvent(ServiceContext)

```
public RecordingTerminatedEvent(javax.tv.service.selection.ServiceContext source)
```

Constructs the event.

Parameters:

source - The `ServiceContext` that generated the event.

Methods

getReason()

```
public int getReason()
```

Returns the reason for the record termination.

This page intentionally left blank.

Annex B org.ocap.dvr.navigation

This annex presents additions to the org.ocap.dvr.navigation package defined in OCAP 1.0.

Package

org.ocap.dvr.navigation

Description

Provides support for Navigation of recording lists.

Class Summary	
Interfaces	
RecordingList	RecordingList represents a list of recordings.
RecordingListIterator	This iterator could be used to traverse entries in a RecordingList.
Classes	
AppIDFilter	Filter to filter based on AppID.
OrgIDFilter	Filter to filter based on OrgID.
RecordingListFilter	Base class for all RecordingListFilters.
RecordingStateFilter	Filter to filter based on values returned by the getState method in org.ocap.dvr.RecordingSession.

org.ocap.dvr.navigation AppIDFilter

Declaration

```
public final class AppIDFilter extends RecordingListFilter
```

```
java.lang.Object
|
+--org.ocap.dvr.navigation.RecordingListFilter
|
+--org.ocap.dvr.navigation.AppIDFilter
```

Description

Filter to filter based on AppID.

Member Summary	
Constructors	
	<pre>AppIDFilter(org.dvb.application.AppID appID)</pre> Constructs the filter based on a particular AppID.
Methods	
boolean	<pre>accept(org.ocap.dvr.RecordingListEntry entry)</pre> Tests if the given RecordingListEntry passes the filter.
org.dvb.application.AppID	<pre>getFilterValue()</pre> Reports the value of AppID used to create this filter.

Constructors

AppIDFilter(AppID)

```
public AppIDFilter(org.dvb.application.AppID appID)
```

Constructs the filter based on a particular AppID.

Parameters:

appID - AppID value for matching RecordingListEntry's.

Methods

accept(RecordingListEntry)

```
public boolean accept(org.ocap.dvr.RecordingListEntry entry)
```

Tests if the given RecordingListEntry passes the filter.

Overrides: accept in class RecordingListFilter

Parameters:

entry - An individual RecordingListEntry to be evaluated against the filtering algorithm.

Returns: true if RecordingListEntry is of the type indicated by the filter value; false otherwise.

getFilterValue()

```
public org.dvb.application.AppID getFilterValue()
```

Reports the value of AppID used to create this filter.

Returns: The value of AppID used to create this filter.

org.ocap.dvr.navigation

OrgIDFilter

Declaration

```
public final class OrgIDFilter extends RecordingListFilter

java.lang.Object
|
+--org.ocap.dvr.navigation.RecordingListFilter
|
+--org.ocap.dvr.navigation.OrgIDFilter
```

Description

Filter to filter based on OrgID.

Member Summary	
Constructors	<pre>OrgIDFilter(org.dvb.application.AppID appID) Constructs the filter based on a particular organization ID.</pre>
Methods	<pre>boolean accept(org.ocap.dvr.RecordingListEntry entry) Tests if the given org.ocap.dvr.RecordingListEntry passes the filter. org.dvb.application.AppID getFilterValue() Reports the value of the organization ID used to create this filter.</pre>

Constructors

OrgIDFilter(AppID)

```
public OrgIDFilter(org.dvb.application.AppID appID)
```

Constructs the filter based on a particular organization ID.

Parameters:

appID - Application identifier containing the Organization ID value for matching org.ocap.dvr.RecordingListEntry instances.

Methods

accept(RecordingListEntry)

```
public boolean accept(org.ocap.dvr.RecordingListEntry entry)
```

Tests if the given org.ocap.dvr.RecordingListEntry passes the filter.

Overrides: accept in class RecordingListFilter

Parameters:

entry - An individual RecordingListEntry to be evaluated against the filtering algorithm.

Returns: true if RecordingListEntry is of the type indicated by the filter value; false otherwise.

getFilterValue()

```
public org.dvb.application.AppID getFilterValue()
```

Reports the value of the organization ID used to create this filter.

Returns: The application identifier containing the organization ID used to filter.

org.ocap.dvr.navigation RecordingList

Declaration

```
public interface RecordingList
```

Description

RecordingList represents a list of recordings.

Member Summary

Methods

	boolean	<code>contains(org.ocap.dvr.RecordingListEntry entry)</code> Tests if the indicated <code>RecordingListEntry</code> object is contained in the list.
<code>RecordingListIterator</code>		<code>createRecordingListIterator()</code> Generates an iterator on the <code>RecordingListEntry</code> elements in this list.
<code>RecordingList</code>		<code>filterRecordingList(RecordingListFilter filter)</code> Creates a new <code>RecordingList</code> object that is a subset of this list, based on the conditions specified by a <code>RecordingListFilter</code> object.
<code>org.ocap.dvr.RecordingListEntry</code>		<code>findRecordingListEntry(javax.tv.locator.Locator locator)</code> Reports the <code>RecordingListEntry</code> corresponding to the specified locator if it is a member of this list.
<code>org.ocap.dvr.RecordingListEntry</code>		<code>getRecordingListEntry(int index)</code> Reports the <code>RecordingListEntry</code> at the specified index position.
	int	<code>indexOf(org.ocap.dvr.RecordingListEntry entry)</code> Reports the position of the first occurrence of the indicated <code>RecordingListEntry</code> object in the list.
	int	<code>size()</code> Reports the number of <code>RecordingListEntry</code> objects in the list.

Methods

`contains(RecordingListEntry)`

```
public boolean contains(org.ocap.dvr.RecordingListEntry entry)
```

Tests if the indicated `RecordingListEntry` object is contained in the list.

Parameters:

`entry` - The `RecordingListEntry` object for which to search.

Returns: `true` if the specified `RecordingListEntry` is member of the list; `false` otherwise.

`createRecordingListIterator()`

```
public org.ocap.dvr.navigation.RecordingListIterator createRecordingListIterator()
```

Generates an iterator on the `RecordingListEntry` elements in this list.

Returns: A `RecordingListIterator` on the `RecordingListEntry`s in this list.

filterRecordingList(RecordingListFilter)

```
public org.ocap.dvr.navigation.RecordingList  
    filterRecordingList(org.ocap.dvr.navigation.RecordingListFilter filter)
```

Creates a new `RecordingList` object that is a subset of this list, based on the conditions specified by a `RecordingListFilter` object. This method may be used to generate increasingly specialized lists of `RecordingListEntry` objects based on multiple filtering criteria. If the filter is `null`, the resulting `RecordingList` will be a duplicate of this list.

Note that the `accept` method of the given `RecordingListFilter` will be invoked for each `RecordingListEntry` to be filtered using the same application thread that invokes this method.

Parameters:

`filter` - A filter constraining the requested recording list, or `null`.

Returns: A `RecordingList` object created based on the specified filtering rules.

findRecordingListEntry(Locator)

```
public org.ocap.dvr.RecordingListEntry  
    findRecordingListEntry(javax.tv.locator.Locator locator)  
    throws InvalidLocatorException
```

Reports the `RecordingListEntry` corresponding to the specified locator if it is a member of this list.

Parameters:

`locator` - Specifies the `RecordingListEntry` to be searched for.

Returns: The `RecordingListEntry` corresponding to `locator`, or `null` if the `RecordingListEntry` is not a member of this list.

Throws:

`javax.tv.locator.InvalidLocatorException` - If `locator` does not reference a valid `RecordingListEntry`.

getRecordingListEntry(int)

```
public org.ocap.dvr.RecordingListEntry getRecordingListEntry(int index)
```

Reports the `RecordingListEntry` at the specified index position.

Parameters:

`index` - A position in the `RecordingList`.

Returns: The `RecordingListEntry` at the specified index.

Throws:

`java.lang.IndexOutOfBoundsException` - If `index < 0` or `index > size() - 1`.

indexOf(RecordingListEntry)

```
public int indexOf(org.ocap.dvr.RecordingListEntry entry)
```

Reports the position of the first occurrence of the indicated `RecordingListEntry` object in the list.

Parameters:

`entry` - The `RecordingListEntry` object for which to search.

Returns: The index of the first occurrence of the entry, or `-1` if entry is not contained in the list.

size()

```
public int size()
```

Reports the number of `RecordingListEntry` objects in the list.

Returns: The number of `RecordingListEntry` objects in the list.

org.ocap.dvr.navigation RecordingListFilter

Declaration

```
public abstract class RecordingListFilter

java.lang.Object
|
+--org.ocap.dvr.navigation.RecordingListFilter
```

Direct Known Subclasses: AppIDFilter, OrgIDFilter, RecordingStateFilter

Description

Base class for all RecordingListFilters. Subclasses of RecordingListFilter can be used to create filters to specify restrictions.

Member Summary	
Constructors	
protected	RecordingListFilter() Constructs the filter.
Methods	
abstract boolean	accept(org.ocap.dvr.RecordingListEntry entry) Tests if a particular entry passes this filter.

Constructors

RecordingListFilter()

```
protected RecordingListFilter()
Constructs the filter.
```

Methods

accept(RecordingListEntry)

```
public abstract boolean accept(org.ocap.dvr.RecordingListEntry entry)
```

Tests if a particular entry passes this filter. Subtypes of RecordingListFilter override this method to provide the logic for a filtering operation on individual RecordingListEntry objects.

Parameters:

entry - A RecordingListEntry to be evaluated against the filtering algorithm.

Returns: true if entry satisfies the filtering algorithm; false otherwise.

org.ocap.dvr.navigation

RecordingListIterator

Declaration

```
public interface RecordingListIterator
```

Description

This iterator could be used to traverse entries in a RecordingList.

Member Summary

Methods

	boolean	hasNext()	Tests if there is a RecordingListEntry in the next position in the list.
	boolean	hasPrevious()	Tests if there is a RecordingListEntry in the previous position in the list.
		nextRecordingListEntry()	Reports the next RecordingListEntry object in the list.
org.ocap.dvr.RecordingListEntry		previousRecordingListEntry()	Reports the previous RecordingListEntry object in the list.
	void	toBeginning()	Resets the iterator to the beginning of the list, such that hasPrevious() returns false and nextRecordingListEntry() returns the first RecordingListEntry in the list (if the list is not empty).
	void	toEnd()	Sets the iterator to the end of the list, such that hasNext() returns false and previousRecordingListEntry() returns the last RecordingListEntry in the list (if the list is not empty).

Methods

hasNext()

```
public boolean hasNext()
```

Tests if there is a RecordingListEntry in the next position in the list.

Returns: true if there is a RecordingListEntry in the next position in the list; false otherwise.

hasPrevious()

```
public boolean hasPrevious()
```

Tests if there is a RecordingListEntry in the previous position in the list.

Returns: true if there is a RecordingListEntry in the previous position in the list; false otherwise.

nextRecordingListEntry()

```
public org.ocap.dvr.RecordingListEntry nextRecordingListEntry()
```

Reports the next `RecordingListEntry` object in the list. This method may be called repeatedly to iterate through the list.

Returns: The `RecordingListEntry` object at the next position in the list.

Throws:

`java.util.NoSuchElementException` - If the iteration has no next `RecordingListEntry`.

previousRecordingListEntry()

```
public org.ocap.dvr.RecordingListEntry previousRecordingListEntry()
```

Reports the previous `RecordingListEntry` object in the list. This method may be called repeatedly to iterate through the list in reverse order.

Returns: The `RecordingListEntry` object at the previous position in the list.

Throws:

`java.util.NoSuchElementException` - If the iteration has no previous `RecordingListEntry`.

toBeginning()

```
public void toBeginning()
```

Resets the iterator to the beginning of the list, such that `hasPrevious()` returns `false` and `nextRecordingListEntry()` returns the first `RecordingListEntry` in the list (if the list is not empty).

toEnd()

```
public void toEnd()
```

Sets the iterator to the end of the list, such that `hasNext()` returns `false` and `previousRecordingListEntry()` returns the last `RecordingListEntry` in the list (if the list is not empty).

org.ocap.dvr.navigation RecordingStateFilter

Declaration

```
public final class RecordingStateFilter extends RecordingListFilter
```

```
java.lang.Object
|
+--org.ocap.dvr.navigation.RecordingListFilter
|
+--org.ocap.dvr.navigation.RecordingStateFilter
```

Description

Filter to filter based on values returned by the `getState` method in `org.ocap.dvr.RecordingSession`.

Member Summary

Constructors

```
RecordingStateFilter(int state)
Constructs the filter based on a particular state type (PENDING, FAILED, etc.)
```

Methods

```
boolean accept(org.ocap.dvr.RecordingListEntry entry)
Tests if the given org.ocap.dvr.RecordingListEntry passes the filter.
int getFilterValue()
Reports the value of state used to create this filter.
```

Constructors

RecordingStateFilter(int)

```
public RecordingStateFilter(int state)
```

Constructs the filter based on a particular state type (PENDING, FAILED, etc.)

Parameters:

`state` - Value for matching the state of a `org.ocap.dvr.RecordingSession` instance.

Methods

accept(RecordingListEntry)

```
public boolean accept(org.ocap.dvr.RecordingListEntry entry)
```

Tests if the given `org.ocap.dvr.RecordingListEntry` passes the filter.

Overrides: `accept` in class `RecordingListFilter`

Parameters:

entry - An individual RecordingListEntry to be evaluated against the filtering algorithm.

Returns: true if org.ocap.dvr.RecordingSession contained within the RecordingListEntry parameter is in the state indicated by the filter value; false otherwise.

getFilterValue()

```
public int getFilterValue()
```

Reports the value of state used to create this filter.

Returns: The value of state used to create this filter.

This page intentionally left blank.

Annex C `org.ocap.media`

This annex presents additions to the `org.ocap.media` package defined in OCAP 1.0.

Package `org.ocap.media`

Class Summary

Interfaces

<code>TimeShiftControl</code>	This interface represents a trick-mode control that can be used for retrieving more information corresponding to the playback of the time-shift buffer.
-------------------------------	---

org.ocap.media

TimeShiftControl

Declaration

```
public interface TimeShiftControl extends javax.media.Control
```

All Superinterfaces: javax.media.Control

Description

This interface represents a trick-mode control that can be used for retrieving more information corresponding to the playback of the time-shift buffer. This control will only be available if the service being presented on the service context is a broadcast service and if there is a time-shift buffer associated with the service context.

Member Summary

Methods

javax.media.Time	getBeginningOfBuffer()	Get the media time corresponding to the current beginning of the time-shift buffer.
javax.media.Time	getEndOfBuffer()	Get the media time corresponding to the end of the time-shift buffer.

Methods

getBeginningOfBuffer()

```
public javax.media.Time getBeginningOfBuffer()
```

Get the media time corresponding to the current beginning of the time-shift buffer. This could be the media time corresponding to start of the buffer, before the buffer wrap around or the media time corresponding to the beginning of the valid buffer area after the wrap around.

Returns: media time corresponding to the beginning of the time-shift buffer.

getEndOfBuffer()

```
public javax.media.Time getEndOfBuffer()
```

Get the media time corresponding to the end of the time-shift buffer. This could be the current system time if the time-shift recording is still ongoing or the media time corresponding to the end point for the valid area of the time-shift buffer.

Returns: media time corresponding to the end of the time-shift buffer.

Annex D org.ocap.storage

This annex presents additions to the org.ocap.storage package defined in OCAP 1.0.

Package

org.ocap.storage

Class Summary	
Interfaces	
AllocateTimeShiftBufferOption	This interface represents a context that can be contained within a StorageProxy and can be used to allocate a time-shift buffer.
MediaStorageOption	This interface represents an option object provided by a StorageProxy that supports media volumes (MediaStorageVolume) that are used by the DVR recording and playback APIs for storing media content.
MediaStorageVolume	This interface represents a media volume on a storage device and is contained within a StorageProxy.
SpaceAllocationHandler	A class implementing this interface decides whether requests to allocate storage space should be allowed or not.
TimeShiftBufferOption	This interface represents a time-shift buffer that can be used to buffer and apply trick modes to a live multi-media stream.
Classes	
ExtendedFileAccessPermissions	This class extends org.dvb.io.persistent.FileAccessPermissions to let granting applications provide read and write file access to applications that have an organization identifier different from a granting application.

org.ocap.storage

AllocateTimeShiftBufferOption

Declaration

```
public interface AllocateTimeShiftBufferOption
```

Description

This interface represents a context that can be contained within a StorageProxy and can be used to allocate a time-shift buffer. This context will only be contained within StorageProxy objects that also contain a MediaStorageContext object. The implementation shall choose a location in the storage context where the buffer can fit.

Member Summary

Fields

```
static int RESET_AT_CHANNEL_CHANGE
    Clear the time shift buffer after every channel change
static int SAVE_AT_CHANNEL_CHANGE
    Keep the contents of the time shift buffer even after channel changes
```

Methods

```
TimeShiftBufferOption allocateMediaBuffer(long size, java.lang.String name, int
    options)
    Allocates a time-shift buffer.
long getMinimumBufferSize()
    Gets the implementation specific minimum time-shift buffer size for this storage
    device.
```

Fields

RESET_AT_CHANNEL_CHANGE

```
public static final int RESET_AT_CHANNEL_CHANGE
```

Clear the time shift buffer after every channel change

SAVE_AT_CHANNEL_CHANGE

```
public static final int SAVE_AT_CHANNEL_CHANGE
```

Keep the contents of the time shift buffer even after channel changes

Methods

allocateMediaBuffer(long, String, int)

```
public org.ocap.storage.TimeShiftBufferOption allocateMediaBuffer(long size,  
    java.lang.String name, int options)  
    throws IllegalArgumentException, OutOfMemoryError
```

Allocates a time-shift buffer. The time-shift buffer shall be added to the `StorageProxy` using the `name` parameter and an appropriate event shall be generated.

Parameters:

`size` - Size of the time-shift buffer in bytes.

`name` - Name of the time-shift buffer.

`options` - time-shift buffer options.

Returns: The `TimeShiftBufferOption` allocated.

Throws:

`java.lang.IllegalArgumentException` - if `size` is too small for the implementation specific minimum buffer size.

`java.lang.IllegalArgumentException` - if the `name` is not unique within the resource registry.

`OutOfMemoryException` - if the storage device does not have enough remaining room for the buffer.

`IOException` - if the allocation will exceed the playback bandwidth of the storage device based on the `@{link MediaStorageContext} getPlaybackBandwidth` method.

`java.lang.OutOfMemoryError`

getMinimumBufferSize()

```
public long getMinimumBufferSize()
```

Gets the implementation specific minimum time-shift buffer size for this storage device.

Returns: Minimum size of an allocatable time-shift buffer in bytes.

org.ocap.storage ExtendedFileAccessPermissions

Declaration

```
public class ExtendedFileAccessPermissions extends
    org.dvb.io.persistent.FileAccessPermissions

java.lang.Object
|
|--org.dvb.io.persistent.FileAccessPermissions
|   |
|   |--org.ocap.storage.ExtendedFileAccessPermissions
```

Description

This class extends `org.dvb.io.persistent.FileAccessPermissions` to let granting applications provide read and write file access to applications that have an organization identifier different from a granting application.

Member Summary

Constructors

```
ExtendedFileAccessPermissions(boolean readWorldAccessRight,
    boolean writeWorldAccessRight, boolean
    readOrganisationAccessRight, boolean
    writeOrganisationAccessRight, boolean
    readApplicationAccessRight, boolean
    writeApplicationAccessRight, int[]
    otherOrganizationsReadAccessRights, int[]
    otherOrganizationsWriteAccessRights)
```

This constructor encodes application, application organization, and world file access permissions as a set of booleans, and other organizations file access permissions as arrays of granted organization identifiers.

Methods

```
int[] getReadAccessOrganizationIds()
    Gets the array of organization identifiers with read permission.

int[] getWriteAccessOrganizationIds()
    Gets the array of organization identifiers with write permission.

void setPermissions(boolean readWorldAccessRight, boolean
    writeWorldAccessRight, boolean readOrganisationAccessRight,
    boolean writeOrganisationAccessRight, boolean
    readApplicationAccessRight, boolean
    writeApplicationAccessRight, int[]
    otherOrganizationsReadAccessRights, int[]
    otherOrganizationsWriteAccessRights)
```

This method allows to modify the permissions on this instance of the `ExtendedFileAccessPermission` class.

Constructors

ExtendedFileAccessPermissions(boolean, boolean, boolean, boolean, boolean, boolean, int[], int[])

```
public ExtendedFileAccessPermissions(boolean readWorldAccessRight,
    boolean writeWorldAccessRight, boolean readOrganisationAccessRight,
    boolean writeOrganisationAccessRight,
    boolean readApplicationAccessRight, boolean writeApplicationAccessRight,
    int[] otherOrganizationsReadAccessRights,
    int[] otherOrganizationsWriteAccessRights)
```

This constructor encodes application, application organization, and world file access permissions as a set of booleans, and other organizations file access permissions as arrays of granted organization identifiers.

Parameters:

`readWorldAccessRight` - read access for all applications

`writeWorldAccessRight` - write access for all applications

`readOrganizationAccessRight` - read access for applications with the same organization as the granting application.

`writeOrganizationAccessRight` - write access for applications with the same organization as the granting application.

`readApplicationAccessRight` - read access for the owner.

`writeApplicationAccessRight` - write access for the owner.

`otherOrganizationsReadAccessRights` - array of other organization identifiers with read access. Applications with an organization identifier matching one of these organization identifiers will be given read access.

`otherOrganizationsWriteAccessRights` - array of other organization identifiers with write access. Applications with an organization identifier matching one of these organization identifiers will be given write access.

Methods

getReadAccessOrganizationIds()

```
public int[] getReadAccessOrganizationIds()
```

Gets the array of organization identifiers with read permission.

Returns: Array of organization identifiers with read permission.

getWriteAccessOrganizationIds()

```
public int[] getWriteAccessOrganizationIds()
```

Gets the array of organization identifiers with write permission.

Returns: Array of organization identifiers with write permission.

setPermissions(boolean, boolean, boolean, boolean, boolean, boolean, int[], int[])

```
public void setPermissions(boolean readWorldAccessRight,  
    boolean writeWorldAccessRight, boolean readOrganisationAccessRight,  
    boolean writeOrganisationAccessRight,  
    boolean readApplicationAccessRight, boolean writeApplicationAccessRight,  
    int[] otherOrganizationsReadAccessRights,  
    int[] otherOrganizationsWriteAccessRights)
```

This method allows to modify the permissions on this instance of the ExtendedFileAccessPermission class.

Parameters:

ReadWorldAccessRight - read access for all applications

WriteWorldAccessRight - write access for all applications

ReadOrganizationAccessRight - read access for organization

WriteOrganizationAccessRight - write access for organization

ReadApplicationAccessRight - read access for the owner

WriteApplicationAccessRight - write access for the owner

otherOrganizationsReadAccessRights - array of other organization identifiers with read access. Applications with an organization identifier matching one of these organization identifiers will be given read access.

otherOrganizationsWriteAccessRights - array of other organization identifiers with write access. Applications with an organization identifier matching one of these organization identifiers will be given write access.

org.ocap.storage MediaStorageOption

Declaration

```
public interface MediaStorageOption
```

Description

This interface represents an option object provided by a `StorageProxy` that supports media volumes (`MediaStorageVolume`) that are used by the DVR recording and playback APIs for storing media content.

The interface distinguishes between content accessible through the DVR APIs and as general purpose files. Implementations may store these different type of content in one or more filesystems. This is transparent to an application. Only the general purpose files are visible through the normal file and directory classes in `java.io`.

The interface can be used to query the amount of storage the storage proxy has for storing all types of application-visible content. (Some of the capacity may be reserved for internal system use.)

The interface also supports the initialization of the storage proxy with a specified allocation between the two types. However, on some implementations, changing the allocations may require filesystems to be destroyed and recreated which may result in the deletion of all application-visible content associated with the storage proxy, including any storage volumes. On other implementations, a change in allocations may require some or all content of the type being reduced to be destroyed. Initialization should be done with extreme caution.

Member Summary

Methods

<code>MediaStorageVolume</code>	<code>allocateMediaVolume(java.lang.String name, ExtendedFileAccessPermissions fap)</code>	Allocates a <code>MediaStorageVolume</code> .
<code>long</code>	<code>getAllocatableMediaStorage()</code>	Gets total allocatable media storage available for all <code>MediaStorageVolume</code> instances.
<code>MediaStorageVolume</code>	<code>getDefaultRecordingVolume()</code>	Gets the default volume that the implementation setup as the default recording volume for the containing <code>StorageProxy</code> .
<code>long</code>	<code>getPlaybackBandwidth()</code>	Gets the playback bandwidth in bits-per-second when only one playback stream and no record streams are open on the entire storage device.
<code>long</code>	<code>getRecordBandwidth()</code>	Gets the record bandwidth in bits-per-second when only one record stream and no playback streams are open on the entire storage device.
<code>long</code>	<code>getTotalGeneralStorageCapacity()</code>	Gets the total capacity of the GPFS available for application use in the storage device.
<code>long</code>	<code>getTotalMediaStorageCapacity()</code>	Gets the total capacity of the MEDIAFS available for application use in the storage device.
<code>void</code>	<code>initialize(long mediafsSize)</code>	Initializes the storage device so that there are at least <code>mediafsSize</code> bytes available for MEDIAFS use.

Member Summary

`boolean simultaneousPlayAndRecord()`
Indicates if the storage device supports simultaneous play and record.

Methods

allocateMediaVolume(String, ExtendedFileAccessPermissions)

```
public org.ocap.storage.MediaStorageVolume allocateMediaVolume(java.lang.String name,
    org.ocap.storage.ExtendedFileAccessPermissions fap)
    throws IllegalArgumentException
```

Allocates a `MediaStorageVolume`. A media volume can contain multi-media content that may impose I/O bandwidth criteria upon the storage device. The new volume will be owned by the application that allocated it.

Parameters:

`name` - Name of the new `MediaStorageVolume`.

`fap` - Access permissions of the new `MediaStorageVolume`.

Returns: Allocated volume storage.

Throws:

`java.lang.IllegalArgumentException` - if the name does not meet Java 1.1.8 directory naming conventions, or if the type is not supported by the storage device.

`java.lang.SecurityException` - if the calling application is unsigned.

getAllocatableMediaStorage()

```
public long getAllocatableMediaStorage()
```

Gets total allocatable media storage available for all `MediaStorageVolume` instances.

Returns: Size of allocatable media storage in bytes.

getDefaultRecordingVolume()

```
public org.ocap.storage.MediaStorageVolume getDefaultRecordingVolume()
```

Gets the default volume that the implementation setup as the default recording volume for the containing `StorageProxy`.

Returns: Default recording volume for the storage device.

getPlaybackBandwidth()

```
public long getPlaybackBandwidth()
```

Gets the playback bandwidth in bits-per-second when only one playback stream and no record streams are open on the entire storage device.

Returns: Playback bandwidth in bits-per-second.

getRecordBandwidth()

```
public long getRecordBandwidth()
```

Gets the record bandwidth in bits-per-second when only one record stream and no playback streams are open on the entire storage device.

Returns: Record bandwidth in bits-per-second.

getTotalGeneralStorageCapacity()

```
public long getTotalGeneralStorageCapacity()
```

Gets the total capacity of the GPFS available for application use in the storage device.

Returns: Total general purpose capacity of the storage device.

getTotalMediaStorageCapacity()

```
public long getTotalMediaStorageCapacity()
```

Gets the total capacity of the MEDIAFS available for application use in the storage device.

Returns: Total audio/video capacity of the storage device.

initialize(long)

```
public void initialize(long mediafsSize)
    throws IllegalArgumentException, IllegalStateException
```

Initializes the storage device so that there are at least `mediafsSize` bytes available for MEDIAFS use. The effects of initialization may include the deletion of all application visible content associated with the storage proxy. Calling this method may remove application access to storage on the device for the duration of the call. It may cause the abnormal termination of applications with open files associated with the storage proxy. This method will block until the storage proxy is again ready for use.

Parameters:

`mediafsSize` - New size of the total MEDIAFS capacity in bytes.

Throws:

`java.lang.IllegalArgumentException` - if the total of the parameters does not equal the value returned by the `getTotalMediaStorageCapacity` method added to the value returned by the `getTotalGeneralStorageCapacity` method.

`java.lang.IllegalStateException` - if the sizes cannot be changed by the implementation for any reason.

simultaneousPlayAndRecord()

```
public boolean simultaneousPlayAndRecord()
```

Indicates if the storage device supports simultaneous play and record.

Returns: True if simultaneous play and record is supported, otherwise returns false.

org.ocap.storage

MediaStorageVolume

Declaration

```
public interface MediaStorageVolume
```

Description

This interface represents a media volume on a storage device and is contained within a `StorageProxy`. A `MediaStorageVolume` is a specialized `LogicalStorageVolume` that supports the recording and playback of media content through the DVR. The volume also provides a mechanism for allocating a fixed amount of storage for use by recordings on the volume.

Member Summary

Methods

<pre>void allocate(long bytes)</pre>	<p>Allocates the specified amount of storage from the containing <code>StorageProxy</code> for use by recordings made to this volume.</p>
<pre>void allowAccess(java.lang.String organizations)</pre>	<p>Adds a list of <code>Organization</code> to the list of <code>Organization</code> who are allowed to use this volume.</p>
<pre>long getAllocatedSpace()</pre>	<p>Gets the amount of space allocated on this volume.</p>
<pre>java.lang.String[] getAllowedList()</pre>	<p>Returns the list of <code>Organizations</code> who are allowed to use this volume.</p>
<pre>long getFreeSpace()</pre>	<p>Gets the remaining available space from an allocation.</p>
<pre>void removeAccess(java.lang.String organization)</pre>	<p>Removes an <code>Organization</code> from the list of <code>Organization</code> who are allowed to use this volume.</p>

Methods

allocate(long)

```
public void allocate(long bytes)
```

Allocates the specified amount of storage from the containing `StorageProxy` for use by recordings made to this volume. The volume is guaranteed to be able to use this amount of storage without requiring the deletion of the contents of other volumes and is also limited to using no more than the allocated amount of storage. The amount of space allocated may be rounded up to meet platform requirements. Once the storage on the volume reaches the amount allocated, the behavior is the same as if the storage device were full, e.g. a `SpaceFullException` is thrown or a `RecordingAlertEvent` generated.

A value of zero indicates that the volume has no minimum guaranteed size and may also use as much space as is available. Until set with the `allocate()` method, the space allocated is zero.

Subsequent calls to `allocate()` change of the existing allocation. However, if a new allocation size is too small to contain existing recordings, a `IllegalArgumentException` is thrown and the allocation size is not changed. Except when the allocation size is changed to zero, which removes the limit and the guaranteed storage size. The allocated space can only be released by an explicit call to `allocate()` or through the deletion of the storage volume.

Parameters:

`bytes` - Number of bytes to allocate.

`bytes` - Number of bytes to allocate.

Throws:

`java.lang.SecurityException` - if the calling application does not have `MonAppPermission("storage")` permission.

`java.lang.SecurityException` - if the calling application does not have `MonAppPermission("storage")` permission.

`java.lang.IllegalArgumentException` - if the requested amount of storage exceeds the amount available for allocation, or reduces the previous allocation making it too small for existing recordings.

allowAccess(String[])

```
public void allowAccess(java.lang.String[] organizations)
```

Adds a list of Organization to the list of Organization who are allowed to use this volume. The volume is owned by the application that created the volume, but is accessible to any record requests where the Organization string matches one of the strings in the organization array.

Parameters:

`organizations` - An array of strings representing organizations that are allowed to use this volume. The String passed as a parameter to the record method should match one of these strings to record onto this volume. If an array of length 0 is passed, any application can use this volume.

Throws:

`java.lang.SecurityException` - if the calling application is not the owner of the volume or does not have `MonAppPermission("storage")` permission.

getAllocatedSpace()

```
public long getAllocatedSpace()
```

Gets the amount of space allocated on this volume. If the allocate method has not been called for, the volume of this method returns 0.

Returns: Number of bytes allocated.

getAllowedList()

```
public java.lang.String[] getAllowedList()
```

Returns the list of Organizations who are allowed to use this volume. The volume is owned by the application that created the volume, but is accessible to any record requests where the Organization string matches one of the strings in the organization array.

Returns: An array of strings representing organizations that are allowed to use this volume.

getFreeSpace()

```
public long getFreeSpace()
```

Gets the remaining available space from an allocation. If no space has been allocated or no allocated space has been used, this method returns the same value as the `getAllocatedSpace` method. Typically, this method will return a value that is changing rapidly.

Returns: Number of bytes available for use from an allocation.

removeAccess(String)

```
public void removeAccess(java.lang.String organization)
```

Removes an Organization from the list of Organization who are allowed to use this volume.

Parameters:

`organization` - A string representing an organization that should be removed from the list of allowed organizations.

Throws:

`java.lang.SecurityException` - if the calling application is not the owner of the volume or does not have `MonAppPermission("storage")` permission.

org.ocap.storage SpaceAllocationHandler

Declaration

```
public interface SpaceAllocationHandler
```

Description

A class implementing this interface decides whether requests to allocate storage space should be allowed or not.

Member Summary

Methods

```
long allowReservation(LogicalStorageVolume volume,  
    org.dvb.application.AppID app, long spaceRequested)  
This method should be used by the implementation to allow the  
SpaceAllocationHandler to grant a request to reserve space.
```

Methods

allowReservation(LogicalStorageVolume, AppID, long)

```
public long allowReservation(LogicalStorageVolume volume,  
    org.dvb.application.AppID app, long spaceRequested)
```

This method should be used by the implementation to allow the SpaceAllocationHandler to grant a request to reserve space.

Parameters:

volume - The LogicalStorageVolume on which the reserved space is requested.

app - The requesting application.

spaceRequested - The new value of the reservation if the request is granted.

Returns: the space granted.

org.ocap.storage

TimeShiftBufferOption

Declaration

```
public interface TimeShiftBufferOption
```

Description

This interface represents a time-shift buffer that can be used to buffer and apply trick modes to a live multi-media stream.

A time-shift buffer may be internal to a storage device and can only be recorded to that device.

Member Summary	
Methods	
void	attach(javax.tv.service.selection.ServiceContext serviceContext) Attaches the time-shift buffer to a javax.tv.service.selection.ServiceContext .
void	detach() Detaches the time-shift buffer from a ServiceContext.
long	getMinimumBufferSize() Gets the implementation specific minimum time-shift buffer size for this storage device.
javax.tv.service.selection.ServiceContext	getServiceContext() Gets the ServiceContext the time-shift buffer is currently attached to.
StorageProxy	getStorageProxy() Gets a StorageProxy that the time-shift buffer is internal or otherwise dedicated to for recording purposes.
long	getTotalCapacity() Gets the total capacity of the time-shift buffer.
long	resize(long newSize) Resizes the time-shift buffer.

Methods

attach(ServiceContext)

```
public void attach(javax.tv.service.selection.ServiceContext serviceContext)
    throws IllegalStateException
```

Attaches the time-shift buffer to a `javax.tv.service.selection.ServiceContext` . If the time-shift buffer is already attached to the `ServiceContext` parameter, this method does nothing. If the time-shift buffer is already attached to a `ServiceContext`, other than the parameter, the `attach` method throws an exception. Otherwise, the time-shift buffer is associated with the `ServiceContext` parameter if the `ServiceContext` argument is valid and is not presenting.

Parameters:

`serviceContext` - The ServiceContext to attach the time-shift buffer to.
`client` - The reserving ResourceClient.

Throws:

`java.lang.SecurityException` - if the caller does not have `ServiceContextPermission("access", "*")`.

`java.lang.IllegalStateException` - if the ServiceContext is in the PRESENTING state, or if the ServiceContext has been destroyed, or if the time-shift buffer is already attached to a ServiceContext different from the parameter, and which is in the PRESENTING state.

detach()

```
public void detach()  
    throws IllegalStateException
```

Detaches the time-shift buffer from a ServiceContext. If not attached this method returns silently.

Throws:

`java.lang.SecurityException` - if the caller does not have `ServiceContextPermission("access", "*")`.

`java.lang.IllegalStateException` - if the ServiceContext is in the PRESENTING state.

getMinimumBufferSize()

```
public long getMinimumBufferSize()
```

Gets the implementation specific minimum time-shift buffer size for this storage device.

Returns: Minimum size of an allocatable time-shift buffer in bytes.

getServiceContext()

```
public javax.tv.service.selection.ServiceContext getServiceContext()
```

Gets the ServiceContext the time-shift buffer is currently attached to.

Returns: The time-shift buffer the ServiceContext is currently attached to. If not attached returns null.

getStorageProxy()

```
public StorageProxy getStorageProxy()
```

Gets a StorageProxy that the time-shift buffer is internal or otherwise dedicated to for recording purposes. If the time-shift buffer is not so dedicated this method shall return null.

Returns: Dedicated storage device or null.

getTotalCapacity()

```
public long getTotalCapacity()
```

Gets the total capacity of the time-shift buffer.

Returns: Total capacity in kilo-bytes.

resize(long)

```
public long resize(long newSize)
    throws IllegalArgumentException, IllegalStateException
```

Resizes the time-shift buffer. A time-shift buffer can only be re-sized if it is not attached to a `javax.tv.service.selection.ServiceContext`, or if the `ServiceContext` it is attached to is not in the PRESENTING state. The implementation may round the size up to a value that is consistent with the storage device configuration.

Parameters:

`newSize` - New size of the time-shift buffer in bytes.

Returns: Actual new size of the time-shift buffer.

Throws:

`java.lang.IllegalArgumentException` - if the `newSize` parameter is smaller than the minimum buffer size or larger than available storage space.

`java.lang.IllegalStateException` - if the time-shift buffer is attached to a `ServiceContext` in the PRESENTING state.